

# Description of Current System

---

"Lizzy Lee's Rock Cafe" is a building in Llangefnih that offers a cafe experience, a shop that sells instruments, music related tools, and they also offer music lessons in a music area below the cafe.

The company was formed in 2020 and hires a small local team mostly consisting of part-timers. Liz Pryde and Pat Lee co-founded the company, with Mrs. Pryde acting as the manager for the Cafe and Music Store, and Mr. Lee managing the music lessons.

Currently the company makes use of paper records but as the company plans for the future they wish to transition over to a digital system for various reasons which will be covered in this document.

## BRIEF OVERVIEW

**Stock Management:** Staff involved in either the Cafe or the Music Shop are required to keep a regularly updated inventory of all stock kept in store. The inventory is used to track sales by the manager.

**Music Lessons:** The teachers use lesson plan sheets to guide their lessons and provide structure for students. They must also record lesson reports including details such as student information, lesson plan used, and the student's presence, behaviour, and any additional notes. The company keeps track of student information such as contact information and music level, and records lesson bookings separately.

**Personal Records:** The company stores records for each of its employees, registered customers, teachers, and students. These records are used to look up contact information or are used for specific cases such as recording an employee's official role in the company.

## MUSIC LESSONS

### **Students:**

Students wishing to be taught in Lizzy Lee's Rock Cafe may first go through the booking process via the company's website, or through physical contact. They are then scheduled to

a lesson as decided by the Lessons Manager, and from there students follow the lesson curriculum.

### **The website booking method is as follows:**

1. The customer may visit the Lizzy Lee's Rock Cafe website.
2. The website contains a page dedicated for booking music lessons with information pertaining to the lessons and the teacher.
3. On this page a customer may fill out a two-step form inquiring their details and an emergency number, and the instrument the student would like to be taught by his/her music teacher.
4. The website sorts the information and sends it to the Manager via email.
5. The email must then be parsed by a human and stored into a register.
6. Pat Lee will follow up on the student's details to assign a lesson plan and teacher.

It should be noted that all students desiring to be taught are required to speak to the Manager of Lessons - Pat Lee - at least once regardless of the method of booking before being accepted as a student..

From here the students can go to their lessons and are able to change their schedules by speaking to their teacher or the manager.

### **Teachers:**

Internally, teachers can view the student schedules and prepare for lessons by studying copies of the lesson plans provided to them by Pat Lee. The lesson plans can be changed by Mr. Lee, and all teachers can access copies of any of them.

The lesson plans include information such as the target level, musical instrument, materials required, objectives of the lesson, and the method of teaching the lesson.

Following each lesson teachers will produce a lesson report describing the student, their presence, what lesson plan was followed, the student's behaviour, and any additional notes relevant to the lesson.

Reports are stored chronologically and can be referred to to assess a student's music comprehension. They are also used by the managers to judge a student's performance in lessons and judge whether or not to move a student up or down a level.

### **Managers:**

Managers use the list of student bookings to calculate the revenue generated from students per month, and use the lesson reports to calculate losses.

Managers are also responsible for organising the schedules to ensure no overlap occurs between lesson times.

## INVENTORY MANAGEMENT:

Inventory management in Lizzy Lee's is a simple task, but is made very time consuming due to the sheer volume of documents recorded each day that build up rapidly with time.

Inventory for both the Music Store and the Cafe are tracked manually by employees. The system requires staff to manually track sales of items on a daily basis, and every other week an employee is required to do a full inventory check to ensure the sales data matches the inventory levels.

When inventory levels fall low Liz Pryde will order new stock, and as shipments arrive an employee will update the inventory to record the change in stock.

## PERSONAL RECORDS:

Employees, teachers, customers, and students that are maintained by the system have records that are stored in paper documents. Each of these records can be updated by a manager, or the person may submit their updated information to a manager where the paper record is replaced.

No backups are kept other than for employees, which are made as they are made or updated. A photocopy is made of the record, and the copy is stored in a regularly accessible location, while the original is placed in a separate locked safe in a separate room.

# Issues and Limitations of the Current System

---

The existing system for Lizzy Lee's Rock Cafe faces some glaring issues that must be addressed when designing the new system that will be replacing it. Many of the issues are caused by the fact that the current system relies entirely on paper records and nothing is stored digitally. Using paper documents poses many risks and has various drawbacks:

## **Time-consuming:**

Managing and updating paper records can be tedious and slow, especially for larger jobs. A digital system can automatically find the required information, freeing up employee time.

## **Inefficient:**

Searching through paper-based systems can waste staff time and slow down production. A digital system can speed up the process of finding specific data, allowing staff to focus on core responsibilities.

## **Lack of organisation:**

Paper records can become disorganised, lost, or damaged, making it difficult to find important information. Digital files are less likely to be misplaced and can't be damaged or lost.

## **Accessibility:**

Accessing paper records requires physical retrieval, which can be difficult for remote staff or those with disabilities. A digital system allows access from any connected device and can include features such as screen readers for greater accessibility.

## **Risk of errors:**

Paper documents are more prone to errors and lack validation checking, increasing the chances of incorrect information. A computer-based system can enforce correct formatting and reduce the chances of errors.

## **Difficulty in data analysis:**

Analysing data from paper records is time-consuming and requires manual compilation. A digital system can automate the process and improve efficiency.

**Lack of security:**

Manually backing up paper documents is time-consuming and backups are often omitted, reducing data security. A digital system can automatically backup data and provide additional security features such as locked folders and recovery options.

**Scalability:**

As the company's record counts grow, most aspects of managing the system will become exponentially more difficult as the number of records to physically sort through become larger. A digital system scales better with the growth of a company's records than a paper system.

**Environmental impact:**

Paper-based systems produce waste and have a negative impact on the environment. Digital systems generate no waste and are more environmentally friendly.

Other limitations of the current system include the impossibility of connecting external processes into the system - e.g. automatically making bookings from a third party stock website whenever inventory runs low, or connecting an electronic till to the inventory book to update levels as sales are made.

# User and system requirements

---

Each user of the system will have different requirements depending on their purpose. Employees will have separate access rights depending on their role, and students should all have equal access rights to their own personal information.

## 1. System Administrators

### a) Music Lessons:

The system administrators will need access to view and add teachers to the system's database, and they will also need to be able to change the cost of customer bookings, including the default price for new customers.

They should also have the ability to review student registration forms from the website and make bookings - setting a time, teacher, instrument, and date according to the information provided. Bookings will be stored and maintained in the new system.

Administrators also require the ability to make calculations from student bookings to estimate the revenue each month from every student.

### b) Music Shop:

Administration needs to have access to view and update the inventory and add stock item records. They should also be able to make calculations using the system's sales calculator to estimate sales over a period of time.

Administrators require a feature to export calculations for revenue according to the information in the sales table.

### c) Personal Records:

Administrators should have full access to all records stored in the system, and be able to change any details. If customers desire to make changes to their record a manager must make the changes for them.

## 2. Music Teachers

### a) Music Lessons:

Teachers should have access to the lesson plan documents and be able to view their own student's bookings.

### b) Music Shop:

Teachers should not have access to the music shop in any way.

### c) Personal Records:

Teachers need to be able to make changes to their own records and see their student's contact information, but should not be able to see any other record information

## 3. Students

### a) Music Lessons:

Students need to be able to view their bookings. If they desire to make changes they should contact their teacher or the manager.

### b) Music Shop:

Students have no need to access the music shop. They can do so by making a customer account.

### c) Personal Records:

Students should be able to change their own contact information

# New System Aims and Objectives

---

The new system would have to contain features for all three parts of the company independently

- In order to achieve this I will create multiple tabs for each section of the company with the required relevant functions for each; Personal Records, Music Lessons, and Inventory Management.
- I will look at how each part of the company runs similarly, and use similar tables so as to avoid recreating tables from the start each time.
- The user will be able to select which section they are dealing with; Cafe, Lessons, or Inventory and the application will change its interface for each selection.

The system should be able to search tables for items and should be able to add new data to tables

- SQLite will be used to create tables and use Key Fields and Foreign Keys to make searching for data easy
- The program should be able to create new data without causing memory overlap or erasing data
- There should be validation in use to prevent incorrect data being stored

The new system will be able to login various users and sort user access

- By using a username and a password the program should be able to allow separate levels of access to users depending on their records.
- Depending on the user the program will be able to restrict or allow access to different parts of the system, i.e. creating a new item in the shop would only be allowed by the admin, accessing sales would be allowed by only the admin, but making sales would be allowed by the members of front staff.
- Only the admin will be allowed to create new stock items and customer records.

## Keep track of stock and manage Out of Date items

- In the cafe any food items will have associated out of date information. The system should be able to automatically determine when these items have gone out of date and warn the user so they can remove the stock
- There can also be an optional feature to warn the users up to 'x' amount of days before an item's out of date to put it on sale.

## Support sales and discounts

- The program will be able to support applying a percentage discount to any item in case of a coupon, or a sale being applied.

# Limitations of new system

---

A new system will be able to solve many limitations of the existing system, but the new system will also be subject to new limitations. These must be addressed as the project is planned and designed

Following are several limitations the new system may have that are immediately apparent:

- Our system will not be able to make use of external libraries for Python, because of this we must make full utilisation of the included libraries and functions of Python to achieve a professional looking and operational application.
- Our application could make use of a calendar system for the student booking timetable. However, we will not be able to make a GUI based calendar widget using Python's built-in libraries. Such a task would require creating an original library to sort dates in a timetable, thus the project must compromise and make use of a less intuitive timetable design.
- The abilities of python without libraries do not allow for expanded internet access, the extent of our abilities in Python concerning the internet may allow us to send emails, but that will be an extensive task and may not be beneficial to our system. Any other web-related tasks will not be possible in our program.
- The program does not have any method of printing or sending files to a printer on every device, instead we will have to export documents to a file for the user to print using an external third party method.
- We are also not able to use full SQL, the only module available to the built in Python library is the SQLite version SQL.
  - SQLite is a serverless database, so it does not need a dedicated server. Because of this the program will store all the database information embedded into itself
- Python is not able to plot graphs without the use of the external module Pandas, which is unavailable to us.
- Python is not able to play video clips in the application, but it is able to play audio clips.

# Feedback

---

**31/08/2022**

- 1) When speaking with Mr Michael Jones about the inventory system I had proposed he suggested I should keep track of sales using a separate table to the inventory and stock items. With this method I would be able to automatically update the inventory as sales are made, and there would also be a record of sales made with date stamps which could be useful to track revenue, and make predictions for sales. I have decided to implement this feature in my design.
- 2) Mr Michael Jones suggested I could improve upon the existing system by implementing a feature that alerts the user of low stock levels automatically as sales are made. This would prevent the potential issue of the existing system of stock running low without the administrators knowing - and cause the customers to have to wait for stock to be delivered. Having this feature would improve the company's stock maintenance system and reduce the risk of running out of stock.
- 3) Initially I was aiming to include an email receipt function in the new system, but after consulting classmate Dominic Williams it was brought to my attention that this would not be possible within the scope of this project due to the fact that I would need to include the smtplib and MIMEText modules which have very limited functionality, and it would require the user to be connected to the server which adds limitations to the system. I have decided to remove this idea from the design.
- 4) Mr Michael Jones advised me to consider the difficulties in implementing a Graphical User Interface incorporating a calendar timetable for the teachers and students. While this feature would be beneficial to both the students and teachers, finding the time to implement such a complex system would be outside the scope of this project, where having a less intuitive interface for the timetables would suffice.

**07/08/2022**

- 1) My classmate Dewi Topps suggested I implement a discount feature into the Sales table instead of making a separate table tracking the discounts that have been applied to sales. This would remove layers of complexity from the program and lower the file count by removing a table - which should speed up the flow of the program. Implementing this feature would also simplify the task of implementing a revenue calculator by removing the need to refer to an external table for each sale.

- 2) Mr Michael Jones suggested I implement a feature where the program will be able to alert the user of upcoming out of date items in the Cafe section. This would be useful in ensuring stock is always fresh and up to date, and also give the user a warning when stock may need to be replaced. This feature should also be able to be turned off by the user if it is not necessary.
- 3) My classmate Dominic Williams suggested I should include a feature in the system which will allow the user to backup and restore data in the system. This would be useful in case of data corruption, and would also be an effective method of making a copy of the data in case it is needed in the future.

## 14/08/2022

- 1) My classmate Dewi Topps suggested I should include a feature that allows the user to view customer history by recording the customer's ID with each sale. This would be useful for tracking customer purchases and analysing their buying habits which can be beneficial for marketing purposes.
- 2) Ms Liz Pryde suggested I should include a feature that tracked sales of customers. This system would be more useful to the company and would simplify the action of tracking stock and sales over a period of time. Initially I had the impression that such a function would be too complicated for the scope of this project, however it is a requirement by the stakeholders of the company so I have decided to include it in the design of the new system.

# Investigation

---

In order to understand the scope of my project, I will investigate the existing system used by the Lizzy Lee's staff. This will require me to undergo interviews and submit questionnaires aimed at all the employees, and further, I will be heading to the office to observe how the system is used.

By doing all of this I will be able to better understand the needs of my system, and I will be able to find ways to improve the current system and expand new features of the existing working system. By the end of this stage of development, I aim to have the required information to create a detailed design for the new system that will cover all aspects required by the stakeholders of Lizzy Lee's.

## INTERVIEW

In order to receive a relevant understanding of the current system I decided to host an interview with the manager, Liz Pryde. For the interview, I prepared a series of questions that would help me better understand how the current system was used and the needs of the current system. I would also ask for ways the current system could be improved upon such as sorting documents by filters automatically instead of having to index manually through pages.

### Interview Questions and Aims

**1. Briefly summarised, what system does the Lizzy Lee's company utilise to manage its stock, employees, and students?**

From this question I hope to understand what exact system the managers and employees use to deal with stock and sort their student's timetables and such. I will also be able to flesh out any parts of the system that is already a requirement that I had not already anticipated.

**2. What are the main requirements covered by your system?**

From this question, I will understand what the company values as necessary for a system, and I will be able to focus on these for my main Aims and Objectives in planning the design of the project.

**3. What users are given access to the current system?**

Depending on the number of users who have access to the system I will be able to decide whether or not to create an entirely different system design for each segment of the system - Shop, Cafe, and Music Lessons - as was brought up in the Feedback of the Discussion on 31/08/2022 // 1, or if it would be a disproportionate amount of work for little use or benefit. I can also decide upon devising more functionality for different users and plan for designing more tables to make up for different types of employees I had not anticipated.

**4. To what level are employees given access to the system?**

This question should help me to understand if, and how the current system manages different levels of users by restricting parts of the system they might have no need for or should not be able to access. Whether or not the system has access rights, or if it is as simple as locking files in a filing cabinet will be helpful information to simulate the same importance of privacy and security.

**5. What part of the system is used most frequently, and what systems do you think need improving on most?**

From this, I aim to have a better understanding of what parts of the system must be streamlined for a better user experience, and also what parts are crucial for designing strongly as they will be tested greatly through practical use. I can also gain insight into what other parts of the system may need improving as a requirement, in contrast to merely migrating the features verbatim to a digital form.

**6. Does the current system make use of backups, and if so how often, and to what degree are the backups kept secure and safe?**

This question will allow me to plan for the possibility to integrate existing backups into the system, and will also give me insight into what preferences the users have for backups. Using a similar schedule would be more beneficial to the company as it would raise the least disruptions for the users and lead to a more streamlined migration - however, if the current backup system is too insecure or could be improved on I will have to weigh the advantages and disadvantages to decide whether or not to change or integrate the current backup system.

**7. What ICT and general computer skills do you, and the team currently have?**

I hope to understand how knowledgeable the users will be in computers. This will underline how far I can go with designing the system technically or will require I include simple navigation in the menus and commands and possibly instructions on how to use the system.

**8. Finally, what new requirements might you and the team have for a new system?**

This will underline the essential improvements Lizzy Lee's desires for a new system that would improve the old system. If any one thing is changed it would have to be what is specified here.

**Transcript of Interview:**

The Manager of Lizzy Lee's insisted on hosting our interview over email, as her ability to meet me in person was disrupted by health conditions. Following is a copy of the text from the email with her signature:

**Briefly summarised, what system does the Lizzy Lee's company utilise to manage its stock, employees, and students?**

1. Currently, at Lizzy Lee's we record a daily inventory that records the stock of our store and the cafe items. We make use of paper documents that record all of our information. When customers are late we refer to a document containing all of their details to contact them, we also keep a log of student attendance.

**What are the main requirements covered by your system?**

2. The system was developed to be able to store the details of our employees, teachers, and students. And to keep track of our stock on a daily basis. It is mainly used for keeping track of the sales we make, and our remaining stock - for both the instruments and music-related products we sell, and the cafe items available at the front of our store.

**What users are given access to the current system?**

3. Access to the current system is primarily given to managers, who are responsible for maintaining and updating records. Teachers have access to student records related to their lessons and can submit progress reports. Employees can access their own records and schedules.

**To what level are employees given access to the system?**

4. Employees have limited access, allowing them to view and update their personal information but not any other personal records. Managers have full access to all records, while teachers have access to student records relevant to their lessons. Front Staff can make and see daily sales, and they can access the customers details. They can also see and change the stock details - at the end of each day we update our inventory of music and cafe items.

Customers and students can make changes to their details by requesting a form, but only have access to their own details.

When they are not being used, the employee, teachers, students, and customer details are locked in a safe in the office that only the Manager can open.

**What part of the system is used most frequently, and what systems do you think need improving on most?**

5. The most frequently used part of the system is the sales and inventory management for the shop and scheduling for music lessons. We need improvements in terms of automating inventory updates, streamlining the scheduling process, and providing easier access to information for employees and teachers.

**Does the current system make use of backups, and if so how often, and to what degree are the backups kept secure and safe?**

6. We currently only maintain hard copy backups of important records such as employee, teacher, student, and customer records. We do store them in a secure location but the backups are only updated every six months due to how time intensive the task is, and I acknowledge it may not be the most secure or efficient solution.

**What ICT and general computer skills do you, and the team currently have?**

7. The team is generally proficient in basic computer skills, such as using office applications, email, and internet browsing. Some employees have experience with more advanced software, but it would be best if the team had access to a design that is more user-friendly and easy to learn.

**Finally, what new requirements might you and the team have for a new system?**

8. A new system would need to have:
  - An intuitive computer based design, otherwise additional training would be required for a complicated computer system.
  - A digital, centralised platform that allows for easy access and updates to records
  - Automation of inventory management and updates
  - Streamlined scheduling for music lessons and employee shifts
  - Different access levels for managers, employees, and teachers
  - Integration of a backup system that is secure and efficient

Sincerely  
Liz Pryde



**Summary of Interview:**

From the interview we can gather the following key points and requirements the manager desires for our digital solution:

**Centralised record management:** The new system should store and manage records for employees, teachers, students, and customers in a digital, centralised platform, making it easy for authorised users to access and update information.

**Inventory management:**

The system should track and automate the process of updating inventory for both music-related products and cafe items. This would streamline daily tasks and reduce manual work.

**Scheduling:**

The new system should provide a streamlined way of scheduling music lessons and employee shifts. This would help the teachers and employees to better manage their time and prevent scheduling conflicts.

**Access levels:**

Different access levels should be implemented for managers, employees, and teachers. Managers should have full access to all records, while teachers should only access student records relevant to their lessons, and employees should only access their personal information and schedules.

**User-friendly design:**

The digital solution should be intuitive and easy to learn, minimising the need for extensive training.

**Backup system:**

The new system should include a secure and efficient backup system, ensuring that important records are safely stored and maintained.

**Compatibility with existing ICT skills:**

The solution should be compatible with the team's current computer skills, such as basic office applications, email, and internet browsing.

However, some needs may not be addressed within the scope of our digital solution:

**Automating stock updates:**

This may not be achievable in the scope of this project. Integrating such a feature would require using a third party module and linking it to another external system to sync sales which would be an astronomical amount of work which is unachievable within our scope.

**Advanced software skills:**

Our solution will be designed with user-friendliness in mind, but it may still require some basic training for employees who are not familiar with using digital systems.

**Streamlined scheduling:**

The new system will aim to include a record of music lessons and will allow for a good representation of the schedules from the information stored, but the system will not be able to include any calendar related features, as there are no modules that can achieve this built into Python - and manually implementing this feature will take a large amount of time that may put it outside the possibilities of this project.

Overall, the digital solution should focus on streamlining and automating processes, improving access to information, and providing a user-friendly interface to meet the needs of Lizzy Lee's management, employees, and teachers.

# DOCUMENT ANALYSIS

In order to obtain information about the existing system I will observe documents, searching for what data is stored and how long it is kept. I will source the documents from various emails sent by the manager with permission from the sales department, or pulled from the [company's website](#).

## Document 1

### **Document Details**

Document number	001
Document Description	Student Booking Invoice
Provided by	Pat Lee (Manager of the Lessons department)
Purpose	To provide students with details of the time schedule and the costs involved. Given to the customer, and stored in records for reference.
Source Format	Email
Frequency	1 created for each student booking
Lifetime	8 years

**Data Fields**

Data	Type	Size	Example	Frequency	Range	Source
Student ID	String	8	80097623	1	0000001-9999999	Added from mail merge
Assigned Teacher Forename	String	20	Harvey	1	N/A	Added from mail merge
Assigned Teacher Surname	String	20	Jones	1	N/A	Added from mail merge
Cost of Lessons per Hour	Integer	2	25	1	1-100	Added from mail merge
Lesson Day	String	9	Wednesday	1	N/A	Added from mail merge
Date of Booking	String	8	LI77 7NZ	1	Date in Past	Added from mail merge

Document 2

**REGISTRATION FORM**

I gofrestru ar gyfer gwersi cerdd, cwbllhewch y ffurflen hon.  
Please complete this form to register for music lessons

Enw / Name *	e-bost/email *	Oed / Age *	Ffon / phone *
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Cyfeiriad / Address *			
<input type="text"/>			
Enw Cysllt brys a rhif ffon / Emergency contact name and phone number *	Perthynas / Relationship *		
<input type="text"/>	<input type="text"/>		
<input type="button" value="Parhau / Continue"/>			

**Document Details**

Item no.	002
Document Description	Student Registration Form
Provided by	Pat Lee (Teacher)
Purpose	To retrieve information of students registration details for booking lessons and further contact
Source Format	Lizzy Lee's website
Frequency	1 created for each registration
Lifetime	8 years

**Data Fields**

Data	Type	Size	Example	Frequency	Range	Source
Student Title	String	5	Mr	1	["Mr", "Mrs", "Ms", "Dr"]	Added from mail merge
Student name	String	20	Harvey Edwards	1		Added from mail merge
Student Age	int	2	16	1	10-99	Added from mail merge
Student email	String	20	harveywdwars@mail.com	1		Added from mail merge
Student Address	Integer	4	19	1		Added from mail merge
Student Contact Name	String	20	Gaynor Edwards	1		Added from mail merge
Student Contact Phone Number	String	11	07726272837	1		Added from mail merge
Student Contact Relationship	String	10	Father	1	["Father", "Mother", "Guardian", "Friend", "Relative"]	Added from mail merge

Document 3**Document Details**

Item no.	003
Document Description	Lesson Plan
Provided by	Pat Lee (Manager of the Lessons department)
Purpose	To set a standard for the lesson structure taught by Music teachers to students of every level
Source Format	Email
Frequency	1 plan kept for each individual plan
Lifetime	10 years (may be submitted for review before then)

**Data Fields**

Data	Type	Size	Example	Frequency	Range	Source
Lesson Title	String	30	Introduction to Guitar Basics	1		Added from mail merge
Lesson Objective	String	50	Students will learn the basic concepts of playing guitar, including proper posture, hand positioning, and basic chords.	1		Added from mail merge
Materials Required	int	2	16	1	10-99	Added from mail merge
Procedure	String	50000	[EXAMPLE BELOW]	1		Added from mail merge
Assessment	Integer	2	9	1		Added from mail merge

**Source (taken from email):**

Lesson Title: Introduction to Guitar Basics

Objective: Students will learn the basic concepts of playing guitar, including proper posture, hand positioning, and basic chords.

Materials:

- Guitars (one for each student)
- Guitar picks (one for each student)
- Handouts with chord diagrams and basic guitar theory

Procedure:

1. Introduction (10 minutes)
  - Greet the students and introduce the lesson objectives.
  - Discuss the importance of proper posture and hand positioning when playing guitar.
2. Hand Positioning (15 minutes)
  - Demonstrate proper hand positioning on the guitar.
  - Have students practise holding the guitar correctly and placing their fingers on the fretboard.
  - Give feedback and correct any errors in hand positioning.
3. Basic Chords (30 minutes)
  - Introduce the basic guitar chords (e.g. C, G, D, E, A) using handouts with chord diagrams and basic guitar theory.
  - Demonstrate how to play each chord and how to transition between chords.
  - Have students practise playing the chords individually and in different combinations.
  - Give feedback and correct any errors in chord playing.
4. Song Application (20 minutes)
  - Choose a simple song that uses the chords covered in the lesson (e.g. "House of the Rising Sun").
  - Demonstrate how to play the song and have students follow along.
  - Break the song down into manageable sections and have students practise playing each section.
  - Put the sections together and have students play the entire song as a group.
5. Wrap-up (5 minutes)
  - Summarise the key concepts covered in the lesson.
  - Answer any questions the students may have.
  - Assign homework (e.g. practice playing the chords and song at home).

Assessment: The lesson will be assessed based on the following criteria:

- The students' ability to follow proper hand positioning and play basic guitar chords.
- The students' ability to transition between chords and play a simple song.

- The quality of the students' participation and engagement in the lesson.
- The teacher's ability to effectively communicate and demonstrate the key concepts of the lesson.

Document 4**Document Details**

Item no.	004
Document Description	Music Item Stock Details
Provided by	Liz Pryde (Manager)
Purpose	To store details of each item stored for the Music Store
Source Format	Email
Frequency	1
Lifetime	Indefinitely (routine checks are made to ensure data is up to date)

**Data Fields**

Data	Type	Size	Example	Frequency	Range	Source
Item ID	String	8	10089112	1		Added from mail merge
Brand	String	50	Gear4Music	1		Added from mail merge
Type	String	50	Amp	1		Added from mail merge
Model	String	50	Roland Cube 80X	1		Added from mail merge
Price	Float	6	213.45	1	0.00-99999	Added from mail merge
Serial Number	String	16	AX011892E	1		Added from mail merge
Date of Purchase	Date	10	12/02/2023	1	Must be date in past	Added from mail merge

Document 5

**Document Details**

Item no.	005
Document Description	Lesson Report
Provided by	Pat Lee (Teacher)
Purpose	To store details of each item stored for the Music Store
Source Format	Email
Frequency	Each Lesson a report is produced
Lifetime	6 months after the student leaves the system

**Data Fields**

Data	Type	Size	Example	Frequency	Range	Source
Lesson ID	String	8	L1001289	1		Added from mail merge
Student ID	String	8	L1001289	1		Added from mail merge
Teacher ID	String	8	L1001289	1		Added from mail merge
Lesson Plan ID	String	8	L1001289	1		Added from mail merge
Student Behaviour	Int	1	2	1	1-4	Added from mail merge
Date	Date	10	12/02/2023	1	Must be date in past	Added from mail merge

# QUESTIONNAIRE

In order to better understand the needs and the potential scope of a new computerised system I decided to design a questionnaire for the current employees of Lizzy Lee's. The questionnaire will inquire about the respondents role in the organisation, and their experience with the current system.

I have decided to use the same questionnaire for all users, and I will be asking for the respondents role in the organisation in order to group answers respectfully. It would be pointless to create a questionnaire for each role as the questions would remain the same for each job type within the company. The results will be grouped depending on the job descriptions the user's provide at the start of the questionnaire.

This questionnaire will be created on Google's 'Forms' service as I will be able to distribute the questionnaire with ease to each employee and it has the added benefit of the simple process of answering digitally on any device with internet capabilities which will invite all the employees to take part in the survey. The Google service also makes analysing the results easier with built in tools such as statistics and graph viewers.

For the questionnaire I have determined a question framework designed to guide my results into a comprehensible format. By doing this my results should be easily categorised and their relevance should be instantly clear.

## Question Layout

1. "What is your current role within Lizzy Lee's?"

I have decided to ask this question to assist with identifying differences in user's experience with the current system depending on their current role in the organisation.

The question will be multiple choice and contain all of the different types of employees hired in Lizzy Lee's:

- Manager
- Sales
- Teacher
- Front Staff

2. "How frequently do you interact with the current system?"

This question will allow me to identify what users, and how many, access the current system frequently, less frequently, and barely at all.

The question will also be multiple choice, with a few degrees of frequency that should cover all potential answers from respondents:

- Multiple times a day
- Once or twice a day
- A few times a week
- A few times a month
- Less than every month
- Never

3. “On a scale of 1 to 10 how well do you understand how the current system is run?”

With this question I hope to be able to gain a better understanding of how the current users of the system have been trained, and potentially how I could try to transfer users' knowledge of the current system to the new system.

The answer will be a scale from 1 to 10, with 1 being indicated as “not understanding”, and 10 being “I understand completely”

4. “How do you feel about using the current system??”

This question will allow me to see feedback on how the respondents like or dislike the current system. From this I can determine whether or not to review the layout and use of the system or to keep the current system's ins and outs.

This question will be a bipolar scale, with 5 options;

- Really dislike
- Slightly dislike
- Neither like nor dislike
- Like
- Really like

5. “How experienced are you with the current system?”

Respondents' answers to this question will reveal the overall fluency of the employees with computers. Depending on these answers I can decide to make use of a Graphics User Interface, or a Command Line Interface. Only if the grand majority of respondents are professionally acquainted with computers should I consider a CLI, otherwise the question will help me to understand how deeply to simplify the system, and how intuitive to make the design.

The question will be a multiple choice question with different levels of fluency of computers:

- I have worked professionally with computers before
- I use computers often but not professionally
- I know how to use a computer

- I am not confident with computers

Following is a link to the final survey: <https://forms.gle/GHQB3mVGjfNmTS1q8>

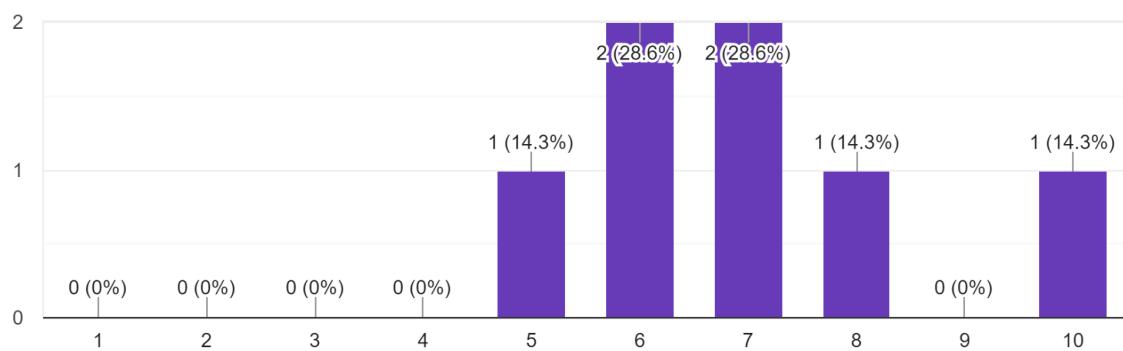
## Response

Lizzy Lee's is a small company with very few employees, so there was a limited number of respondents. This limits the accuracy of the survey results, as outliers will have a bigger impact on the results, but despite this the results still prove to be insightful and can be used to determine how I will design the new system.

Looking at the results it is apparent that the majority of users somewhat understood how the current system works, with one having full comprehension. This is good news as transitioning to a new system may not be such a difficult transition considering their weak feelings toward how the current system is ran:

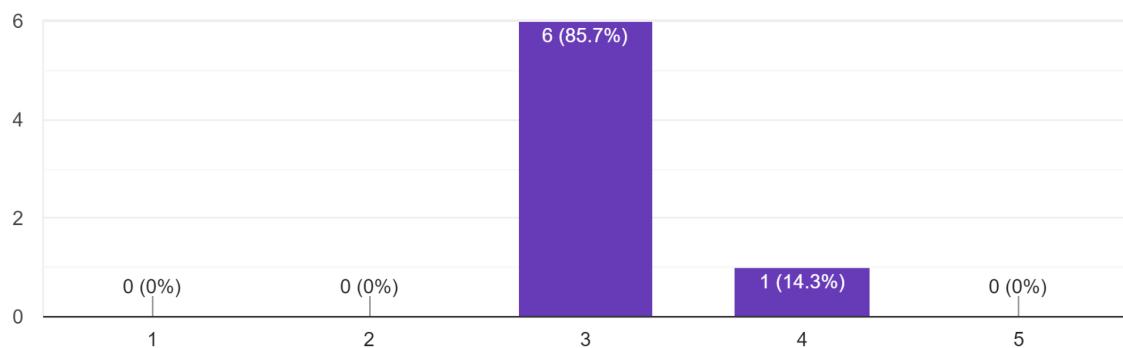
On a scale of 1 to 10 how well do you understand how the system is run?

7 responses



How do you feel about using the current system?

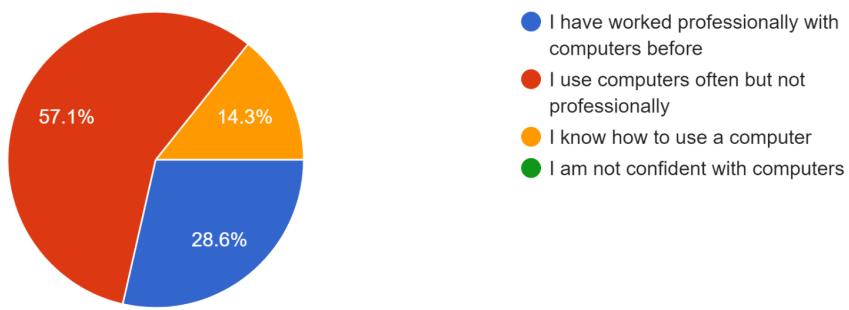
7 responses



A majority of the respondents claimed that they were capable with computers, although not many were professionally trained; this has helped me make the decision to use a Graphical User Interface (GUI) in place of a Command Line Interface (CLI), and has also reminded me to keep the design of the GUI simple for all users to benefit from.

## How experienced are you with computerized systems?

7 responses



I could also gather from the results that the Sales team were experienced with computers professionally, meaning that the Sales users would be able to use a more complicated design incorporating filters and data structures. Therefore I could allow the Sales team to use a more complicated interface to speed up their workflow and increase productivity, while preventing the same interface from slowing down less experienced users.

# Desk Research

Since the new system aims to implement a digital solution for; managing lessons, student progress, managing employees, and stock recording - desk-based research into existing software such as Learning Management System (LMS), and Employee Management and Human Resources (HR) software will be valuable. This research will provide insights into the features and functions commonly found in these systems and how they can be adapted to work effectively for Lizzy Lee's. The observations made in this section will be considered for implementing the final design.

## Learning Management System - Moodle

I decided to examine Moodle as an example of a Learning Management System (LMS) software to gain an understanding of its features and user experience. Moodle is a widely-used open-source LMS that offers various tools for managing courses, assignments, and tracking student progress. Some key features and functionalities that can be adapted for Lizzy Lee's system include:

**Course creation and management:** Moodle allows teachers to create and organise courses designed to cover a range of skill levels. This feature can be adapted to manage music lessons and course materials for Lizzy Lee's.

**Submission and grading:** Moodle enables teachers to provide feedback on student performance for each lesson. This functionality can be used to track student progress, lesson completion, and teacher feedback in the new system.

**Calendar and scheduling:** Moodle includes a calendar feature that displays upcoming events, and lesson schedules. A similar calendar interface would be very beneficial to help students and staff at Lizzy Lee's to plan and organise lessons, and other events.

## Employee Management and Human Resources - Gusto

I also looked into Gusto as an example of Employee Management and Human Resources (HR) software to understand its features and how they can be applied to Lizzy Lee's Rock Cafe's employee management needs. Gusto is a cloud-based HR platform that offers tools for managing employee records, payroll, benefits, and time tracking. While the diverse level of rich features from this software high outranks the requirements of Lizzy Lee's, there are still many key features and functionalities that can be incorporated into the new system, which include:

**Employee profiles and records:** Gusto allows managers to create and update employee profiles, which can include personal information, contact details, and job-related data. This feature can and must be adapted to manage records for employees, teachers, and students

in the new system, as it is necessary to store details of employees and keep them up-to-date

**Time tracking and attendance:** Gusto offers a time tracking feature that enables employees to log their work hours, request time off, and view their schedules. Tracking logged hours, as well as student lesson attendance is a functionality that can be implemented to make estimating revenue and costs easier with a computer system.

By examining the features and functionalities of Moodle and Gusto, valuable insights can be gained into how these solutions can be adapted to create an effective digital system for Lizzy Lee's Rock Cafe. The information gathered will be used to design and implement a system that meets the needs of students, teachers, employees, and management.

# Stakeholders

During my investigation I was contacted by the stakeholder Elwyn Hughes who was interested in the use of a computer system to store data in a database. Specifically, he was keen on how storing data digitally would improve the efficiency of accessing stock data, and keeping an inventory.

Elwyn Hughes is a veteran employee who has worked with the company since their beginnings. He works primarily in the front of the store, and makes frequent daily use of the stock system. Each day he checks the stock with the sales to ensure they match up - which is a very taxing task. With a computer system the company would be able to compare daily sales and stock differences for easier analysis of data.

After speaking to Elwyn Hughes I approached users from different departments and areas inquiring about their requirements. Following is a list of demands by members of each level within the company:

## Manager

The Managers require a system that can:

- Access all data from any part of the system
- Utilise a log-in system to separate user departments
- Create or delete users from the system
- Make changes to student/employee/teacher tables and edit specific fields
- Have all functionality of the system available to them
- Restrict functions from users who should not have access to them - such as restricting the sales team from accessing sensitive student data
- View customer reports and student registration forms

## Front Staff / Cashier

The Front staff require a system that can:

- Record sales of cafe items and musical products
- Only view customer data
- View and edit stock details for both cafe items and music products

## Teachers

Teachers require a system that can:

- Log in to separate accounts to view associated student details
- View student bookings
- Change their availability

- View their timetable
- Receive/review student registration forms

## Students

Students require a system that can:

- Log in separate accounts to view their own custom bookings
- Change their availability
- Change their instrument and request to change teacher or drop out of lessons

## Customers

Customers require a system that can:

- Keep record of purchases

# Inputs and Outputs of Current System

## INPUTS

From all aspects of the Investigation process I have accrued a list of all the inputs required by each process the company has to record. This list will prove to be crucial as part of the development stage of the process as I will be able to refer back to these inputs to decide what I should include in the user interface, as well as the data structure I choose to use.

**Process:** Student Registration

**Inputs:**

- Forename
- Surname
- Sex
- Title
- Birthdate
- Town
- County
- Postcode
- Guardian Contact Phone Num
- Phone Num
- Email

**Process:** Employee Hire

**Inputs:**

- Forename
- Surname
- Sex
- Title
- Birthdate
- Hire Date
- Job Description
- Town
- County
- Postcode
- Phone Num
- Email

**Process:** Teacher Hire

**Inputs:**

- Forename
- Surname
- Sex
- Title
- Birthdate
- Hire Date
- Town
- County
- Postcode
- Phone Number
- Email

**Process:** Customer Registration

**Inputs:**

- Forename
- Surname
- Sex
- Title
- Birthdate

- Town
- County
- Postcode
- Phone Num
- Email

**Process:** Student Booking**Inputs:**

- Student
- Teacher
- Location
- Lesson Frequency
- Lesson Day
- Lesson Time

- Lesson Length
- Lesson Cost
- Booking Date
- Booking Update
- Cancelled

**Process:** Stock Update**Inputs:**

- Brand
- Type
- Model

- Price
- Serial Number
- Date of Purchase

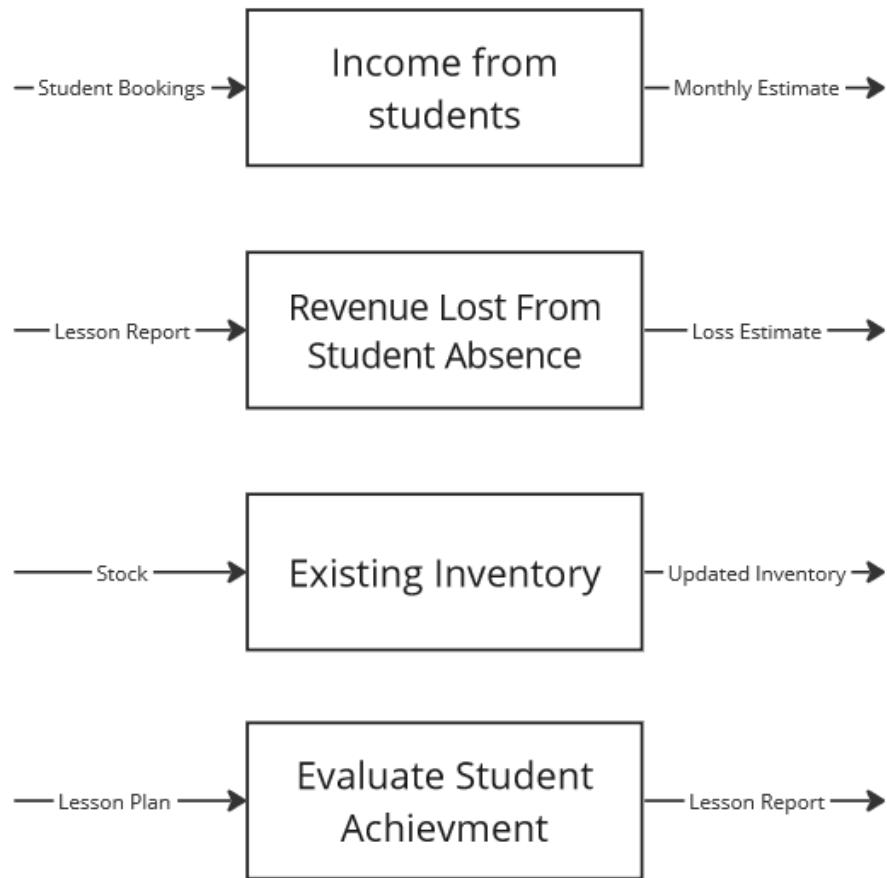
**Process:** Create Lesson Plan**Inputs:**

- Lesson Title
- Lesson Objective
- Materials Required

- Procedure
- Assessment

# OUTPUTS

As observed during my Investigation, the current system does not only accept data input, but also requires the processing of data to produce various new documents required by members of staff. The documents required to create an output are as follows:



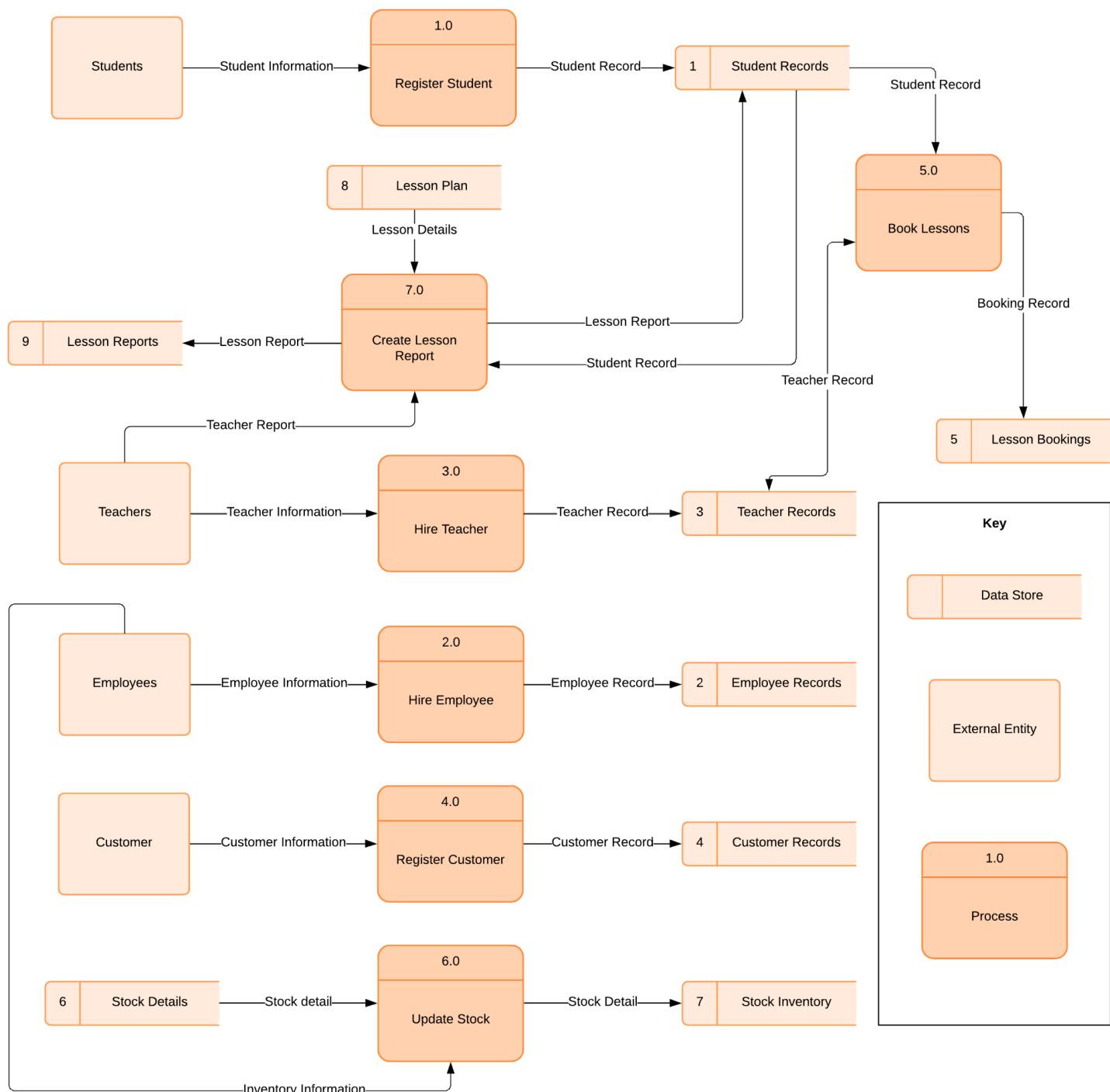
## UPDATES

I also identified the following records being updateable as a requirement in order to keep their data up-to-date:

- Employee Details
- Teacher Details
- Student Details
- Customer Details
- Student Bookings
- Stock Details
- Stock Inventory

# Data Flow

Using the previous information in the Inputs and Outputs of Current System section I have created this Data Flow Diagram to highlight the flow of data in the current system. This information will prove crucial in the development stage when designing the new system's interface and data transfer and storage.



# The Limitations of the Existing

The current system is entirely stored in paper documents. According to Liz Pryde this has led to issues of management and storage. Here is a full list of issues noted by Liz Pryde during my investigation:

As a result of the paper based system there are many issues with the current system that can be easily fixed with a digital upgrade. These are limitations that stem from the current system's function that I hope to remedy with the new system:

## **Time Consuming:**

Having to manage and update paper records can take time and be tedious for larger batch jobs, and can especially be frustrating on employees having to search for a specific piece of information. Using a digital system will free up employee time by automatically finding the required information.

## **Inefficient:**

Similar to the aforementioned issue, having to index through paper based systems prevents staff from spending time on important tasks which slows down production of the team. Core responsibilities can be freed up by freeing time using a digital system that can speed up the process of finding specific data

## **Lack of Organisation:**

Paper records can easily become disorganised because of damage or being misplaced. This can lead to further difficulties in finding important information. Having files stored digitally prevents misplacing information in a concealed environment, and digital files can not be damaged or lost solving this shortcoming in design.

## **Accessibility:**

The paper records that are required can only be accessed by physically going to the location where they are stored and retrieving them. This can lead to issues with staff requiring remote access, or even prove to restrict access to records from members of staff with disabilities preventing them from moving frequently, or preventing them from physically reading paper. A digital system solves both these limitations by allowing access to files from any node connected to the system and can also make documentation more accessible with features such as Screen Reader.

## **Risk of Errors:**

Paper has a higher chance of receiving errors as there is no validation checking when documents are recorded, and reliability is reduced as all information could potentially be incorrect, whereas a computer based system would allow for the system to enforce data to be of a correct type and format to reduce chances of errors.

**Difficulty in Data Analysis:**

Analysing data from paper records is challenging and time consuming. Data has to be manually compiled and organised to make an analysis which is tedious. A computer would be able to take the data stored in its system and make analysis decisions itself, or improve the Data Analysts efficiency with digital tools.

**Lack of Security:**

Manually backing up paper documents is extremely time consuming, and vastly increases the number of resources used by the system. Because of this backups are usually omitted and so the security of data is reduced, or if backups are made and destroyed, there is no method of data recovery. Physical data can also be stolen, lost, or damaged. A digital system would be able to make automatic backups, and having backups only increases resource use slightly compared to with a paper based system. There are also the added benefits of having locked folders, backup recovery, and no physical way of being stolen adding to the security of a digital system.

**Environmental Impact:**

A paper based system also produces a large impact on the environment, with large numbers of waste being generated as the company grows. Digital systems do not generate waste, and so have a lesser effect on the environment.

# Full Specification of New System

## SUMMARY OF PURPOSE OF PROJECT

The purpose of this project is to develop a user-friendly and efficient digital system to manage Lizzy Lee's Rock Cafe operations, including; managing customer, employee, teacher, and student records, inventory management, and managing student lessons. The system will streamline the existing paper-based process, enabling easier access to information and reducing manual data entry and access. Different levels of access will be implemented, allowing managers, employees, teachers, and students to interact with the system according to their roles and permissions.

## METHOD

To build the system, Python 3 will be used as the programming language, given its versatility and the wide range of available libraries. The system will be organised into separate modules, which will improve maintainability and simplify navigation through the codebase. The following libraries will be employed in the development process:

1. tkinter: This library will be used to create the graphical user interface (GUI) of the application, enabling users to interact with the system in an intuitive manner.
2. sqlite3: This lightweight database management system will be responsible for handling all data storage and retrieval tasks. SQLite will be used to store customer, employee, teacher, and student records, inventory details, and lesson information.
3. datetime: This library will be utilised to manage date and time-related information, such as lesson schedules and employee work hours, ensuring accurate and efficient data handling.
4. csv: This library will be required to parse files of the csv file type and improves functionality of storing to csv files for exporting stock and lesson data in a structured file, and improves the app's ability to batch import data from csv files.
5. re: The Regular Expression library is vital for utilising validation techniques for all input fields. The library also supports custom expression definitions allowing the system to support custom validation methods.

# FULL SPECIFICATION OBJECTIVES

## **1. User Access and Login System:**

1.1 - Restrict app to allow different levels of access for managers, employees, teachers, and students.

1.2 - Login system to authenticate users and grant appropriate permissions. Users will only have access to the features of the GUI their access levels allow.

## **2. Customer Records:**

2.1 - Managers can create, update, and delete validated customer records with their personal information.

2.2 - Display list of customers with personal information to managers in a searchable and sorted table.

2.3 - Display the purchase history of customers in a searchable and sorted table.

## **3. Employee Records:**

3.1 - Managers can create, update, and delete validated employee records with their personal information.

3.2 - Display list of employees with personal information to managers in a searchable and sorted table.

3.3 - Employees can edit their own records.

## **4. Teacher and Student Records:**

4.1 - Managers can create, update, and delete validated teacher and student profiles with their personal information.

4.2 - Display list of teachers and students with personal information to managers in a searchable and sorted table.

4.3 - Teachers and students can edit their own records

4.4 - Teachers and managers can view the progress and attendance of students in their respective lessons.

## **5. Sales and Inventory Management:**

5.1 - Managers can create, update, and delete validated stock items for both the music store and the cafe with detailed information, such as product name, description, and price.

5.2 - Record inventory levels over a period of time.

5.3 - Track sales from user input.

5.4 - Automatically update inventory levels with sales.

5.5 - Allow managers to add items to inventory levels.

## **6. Student Lessons:**

6.1 - Managers can create, update, and delete each student's lesson bookings.

6.2 - Managers and teachers can view searchable and sorted lists of student reports.

6.3 - Teachers can submit lesson reports after each lesson.

6.4 - Managers can create lesson plans.

6.5 - Managers and teachers can view a searchable and sorted list of lesson plans.

### **7. Reporting and Data Analysis:**

- 7.1 - Calculate and display sales report given a date range.
- 7.2 - Calculate and display value of inventory.
- 7.3 - Display estimate of revenue from student lessons.
- 7.4 - Support exporting calculations to .txt or .csv format.

### **8. Data Security and Integrity**

- 8.1 - Encrypt all stored data to prevent personal information being leaked.
- 8.2 - Make routine backups of all stored data.
- 8.3 - Validate all user input.

By implementing these features, the digital system for Lizzy Lee's Rock Cafe will significantly improve the efficiency and accuracy of the business operations, eliminating the need for paper-based processes and providing a more streamlined, organised, and accessible solution for managing the cafe's various requirements.

Liz Pryde was a valuable asset in creating the specification. She ensured the final objectives aligned with the requirements of the company in all aspects, and that the new system could scale with the future in mind.

# Design

---

## FULL SPECIFICATION OBJECTIVES

### **1. User Access and Login System:**

- 1.1 - Restrict app to allow different levels of access for managers, employees, teachers, and students.
- 1.2 - Login system to authenticate users and grant appropriate permissions. Users will only have access to the features of the GUI their access levels allow.

### **2. Customer Records:**

- 2.1 - Managers can create, update, and delete validated customer records with their personal information.
- 2.2 - Display list of customers with personal information to managers in a searchable and sorted table.
- 2.3 - Display the purchase history of customers in a searchable and sorted table.

### **3. Employee Records:**

- 3.1 - Managers can create, update, and delete validated employee records with their personal information.
- 3.2 - Display list of employees with personal information to managers in a searchable and sorted table.
- 3.3 - Employees can edit their own records.

### **4. Teacher and Student Records:**

- 4.1 - Managers can create, update, and delete validated teacher and student profiles with their personal information.
- 4.2 - Display list of teachers and students with personal information to managers in a searchable and sorted table.
- 4.3 - Teachers and students can edit their own records
- 4.4 - Teachers and managers can view the progress and attendance of students in their respective lessons.

### **5. Sales and Inventory Management:**

- 5.1 - Managers can create, update, and delete validated stock items for both the music store and the cafe with detailed information, such as product name, description, and price.
- 5.2 - Record inventory levels over a period of time.
- 5.3 - Track sales from user input.
- 5.4 - Automatically update inventory levels with sales.

**6. Student Lessons:**

- 6.1 - Managers can create, update, and delete each student's lesson bookings.
- 6.2 - Managers and teachers can view searchable and sorted lists of student reports.
- 6.3 - Teachers can submit lesson reports after each lesson.
- 6.4 - Managers can create lesson plans.
- 6.5 - Managers and teachers can view a searchable and sorted list of lesson plans.

**7. Reporting and Data Analysis:**

- 7.1 - Calculate and display sales report given a date range.
- 7.2 - Calculate and display value of inventory.
- 7.3 - Display estimate of revenue from student lessons.
- 7.4 - Support exporting calculations to .txt or .csv format.

**8. Data Security and Integrity**

- 8.1 - Encrypt all stored data to prevent personal information being leaked.
- 8.2 - Make routine backups of all stored data.
- 8.3 - Validate all user input.

# ABSTRACTION

The upgraded digital system for Lizzy Lee's Rock Cafe will have to consider many aspects of digital safety while covering a broad range of features required by the company. In this section each function required by the stakeholders will be explained in detail, I aim to produce enough detail that the software development stage will be as simple and error-free as possible.

To accomplish a comprehensive design I will thoroughly explain the final design and layout of the application and I will have included a list of every piece of data that the system will store including how this data will be validated for security.

I will also create pseudocode designs for algorithms the system will use to ensure that as I develop the software it will run efficiently with well designed algorithms, and I can understand the purpose of each algorithm.

This design document should be able to guide any third party developer to understand how the final system should look and function, and any team should produce a similar result that will be usable by the Lizzy Lee's Rock Cafe team.

# PROCESSES

The system will make use of individual and separate processes that will accept certain inputs and display specific outputs for the appropriate users depending on the function of the process. Here I will layout the exact purpose and flow of each process:

## User Access and Login System

- **Login Screen:** Users will be presented with a login screen to enter credentials (username and password). The system will automatically authenticate their input and grant specific access depending on their assigned role in the system - Manager, employee, teacher, or student.
- **User Dashboard:** Successfully logging in will direct users to a dashboard where they can access relevant information and their available functions.

## Customer Records

- **Customer List:** Users with the appropriate permissions will have access to a full list of customers, displaying customer details such as their name, contact details, and a link to their purchase history. The user will be able to add to these records and will be able to update or delete existing records.

## Employee Records

- **Employee List:** Managers will have access to a full list of employees, displaying employee details such as their name, contact details, and job description. The user will be able to add to these records and will be able to update or delete existing records.

## Teachers and Student Records

- **Teacher List** - Users with appropriate permissions will have access to a full list of teachers including their basic information such as name, contact details, and assigned lessons. The user will be able to add to these records and will be able to update or delete existing records.
- **Student List** - Users with appropriate permissions will have access to a full list of students, displaying their basic information such as name, contact details, assigned lessons, and behaviour and grade metrics. The user will be able to add to these records and will be able to update or delete existing records.

### Inventory Management

- **Stock List** - Users with appropriate permissions will be able to view and create stock items in a list. Stock records will include the name of the item, its type, price, data acquired, and so on. The stock items are to be referenced in the inventory. The user will be able to add to these records and will be able to update or delete existing records.
- **Inventory** - Users with appropriate permissions can view and update the inventory records. This will include data such as stock ID, date stored, and count. The user will be able to add to these records and will be able to update or delete existing records.
- **Sales** - Users with appropriate permissions can add a sale to the system. Sales will record all the music, and cafe items being sold and any discounts applied and have the ability to record the ID of the customer initiating the sale.

### Student Lessons

- **Lesson Bookings** - Users with appropriate permissions will be able to view a list of bookings students have made with teachers. Bookings will include information such as the student, teacher, lesson frequency, and cost. The user will be able to add to these records and will be able to update or delete existing records.
- **Lesson Plans** - Users with the appropriate permissions will be able to view, edit and add lesson plans for teachers to use in their lessons. Plans will outline the target student grade, purpose of the lesson, materials required, and procedure. The user will be able to add to these records and will be able to update or delete existing records.
- **Lesson Report** - Teachers will be able to submit a user report after each lesson where they can link the lesson plan used, and grade the students achievements, and behaviour throughout the lesson. The user will be able to add to these records and will be able to update or delete existing records.

### Data Export

- **Lesson Plan Printout** - Teachers will be able to print out the lesson plans in order to prepare lessons for their students. This will produce a text file of the required text file.
- **Cost Estimates** - Users with the appropriate permissions will be able to produce a computer generated estimate of income from students, loss from student absences, and inventory value.
- **Booking Printout** - Users with appropriate permissions will be able to print out a copy of a student's booking, containing information such as the costs, lesson time, teacher, and instrument.

### Data Exploration

- **Filter Records** - Users with view access to records will be able to filter the displayed records by giving specified filters - such as matching details, specific date range, or price range.

- **Sort Results** - To speed up the process of looking for specific records, users will be able to sort results by alphabetical order, date, or number value for specific columns in records.

### Backups

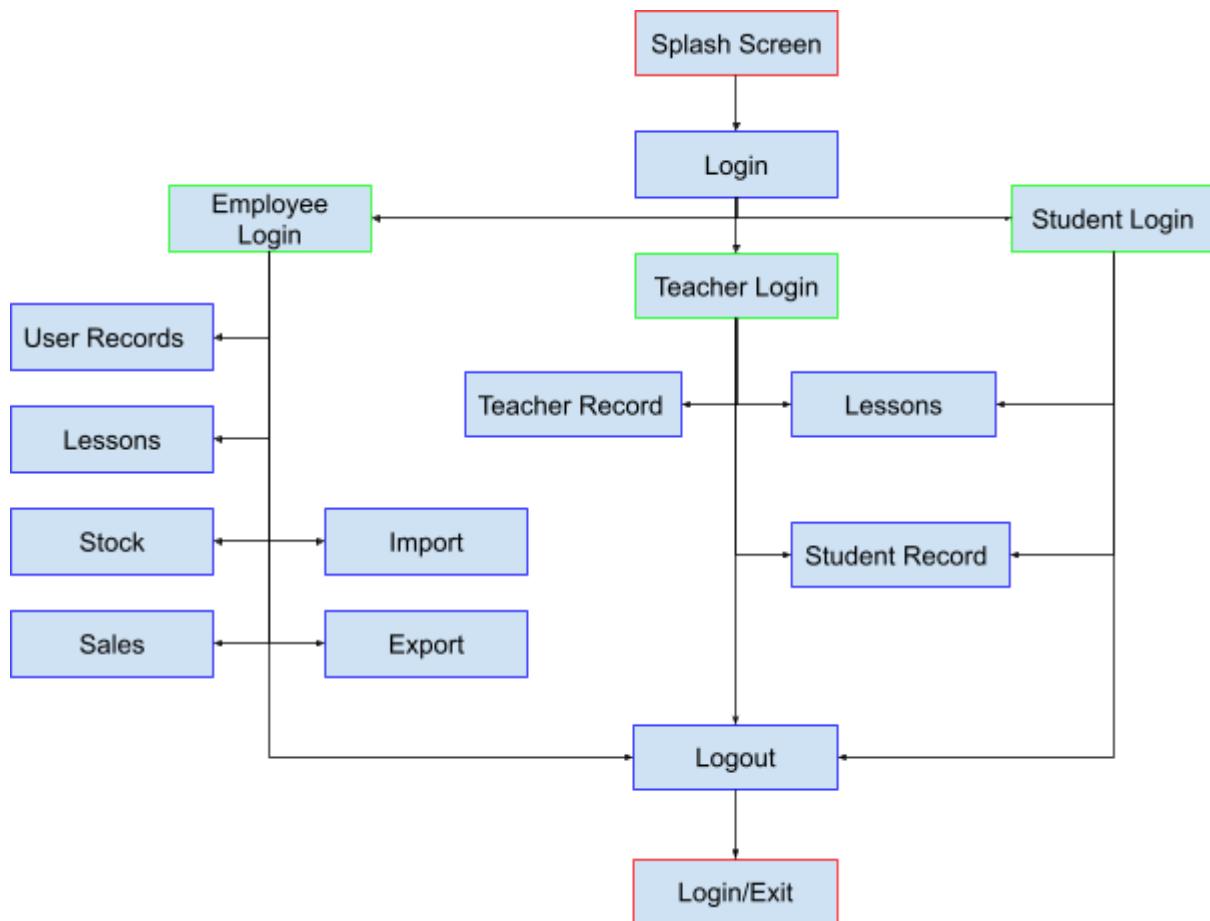
- **Automatic Backups** - The system should automatically make backups every day in a folder.
- **Backup Browsing** - Managers should be able to browse old backups to view their data, but not make changes to them.

By breaking down the system into these processes the final product should follow a comprehensive and user-friendly solution for managing all aspects of the business. By employing a modular design users should be able to navigate their relevant information and functions with ease, and users with limited understanding of computers should be able to intuitively control the program with limited training.

# Program Flow

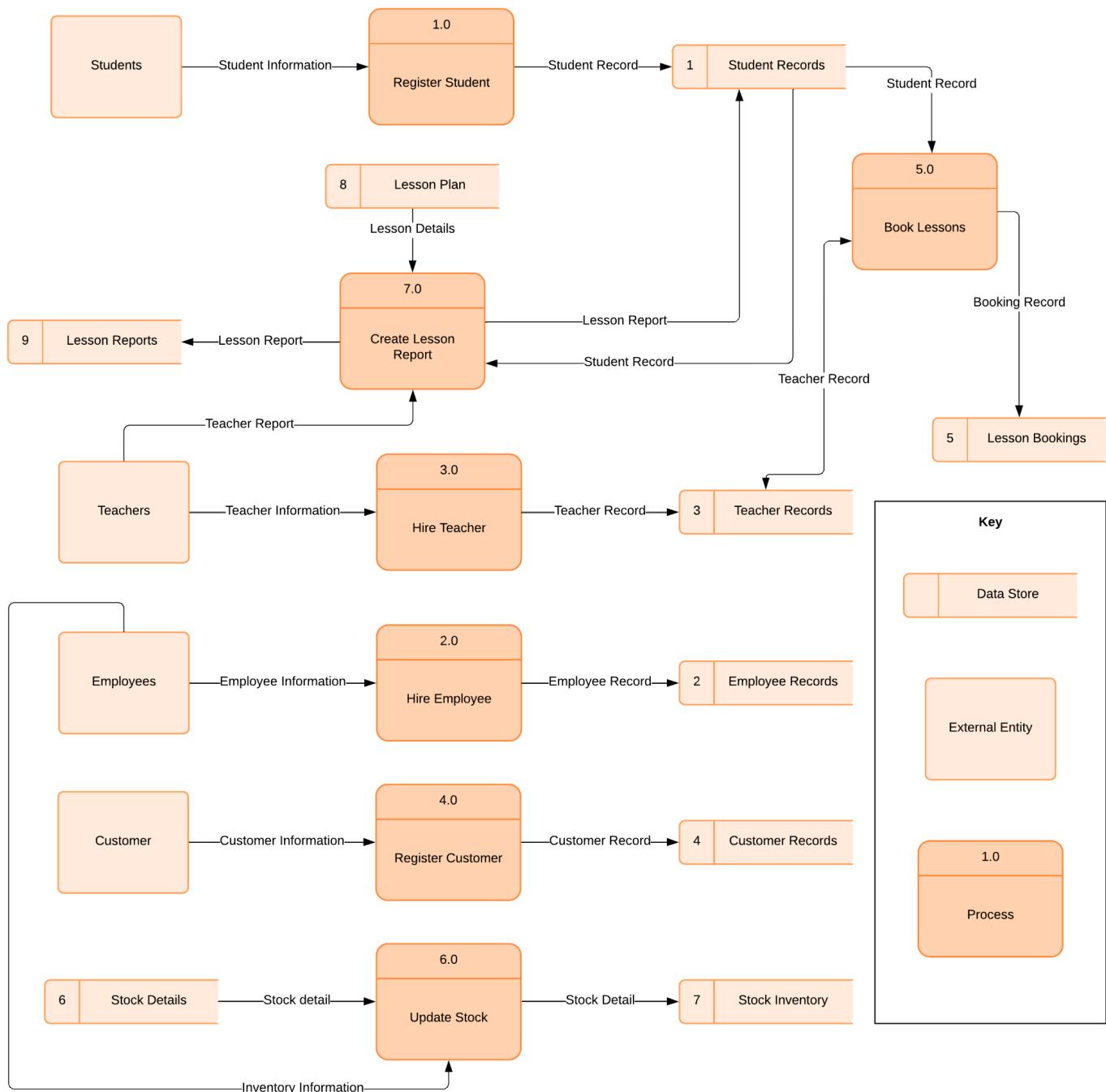
## User Flow Tree Diagram

The following user flow diagram will be used in production to create a system that provides users with a natural flow from screen to screen. using this system the program will be intuitive and user friendly.



# Data Flow Diagram

The data flow diagram demonstrates how data will flow between user input and system processes.



# Entity Relationship Diagram

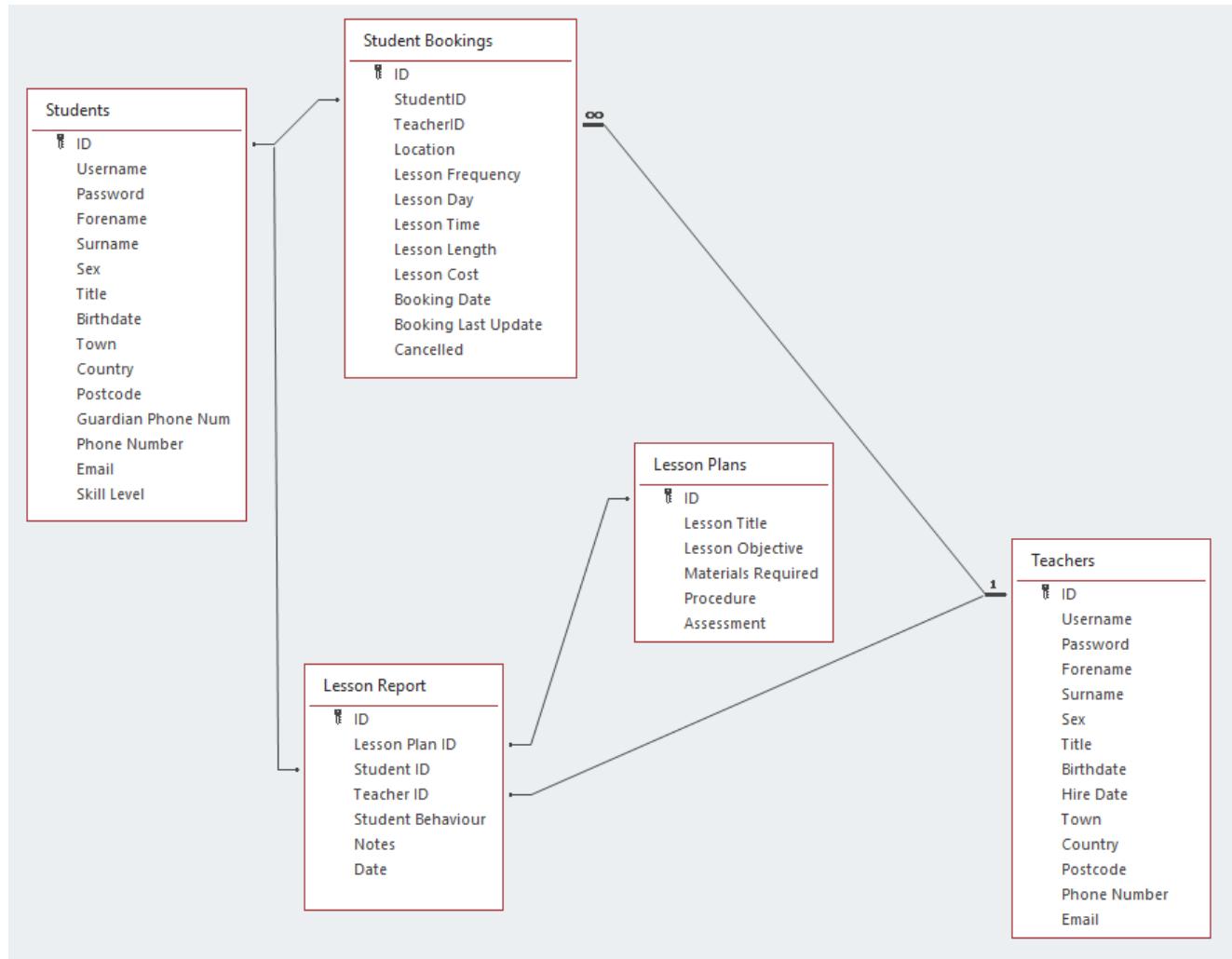
An Entity Relationship Diagram will represent how tables will interact with each other. Nodes can show table connections with different types of relationships; one-to-one - represented by two unmarked points, and one-to-many - represented by a '1' point connected to a '∞' point.

## Student Bookings:

Students and teachers will be linked through the student bookings table. Every student must have one booking, and each teacher can have many student bookings.

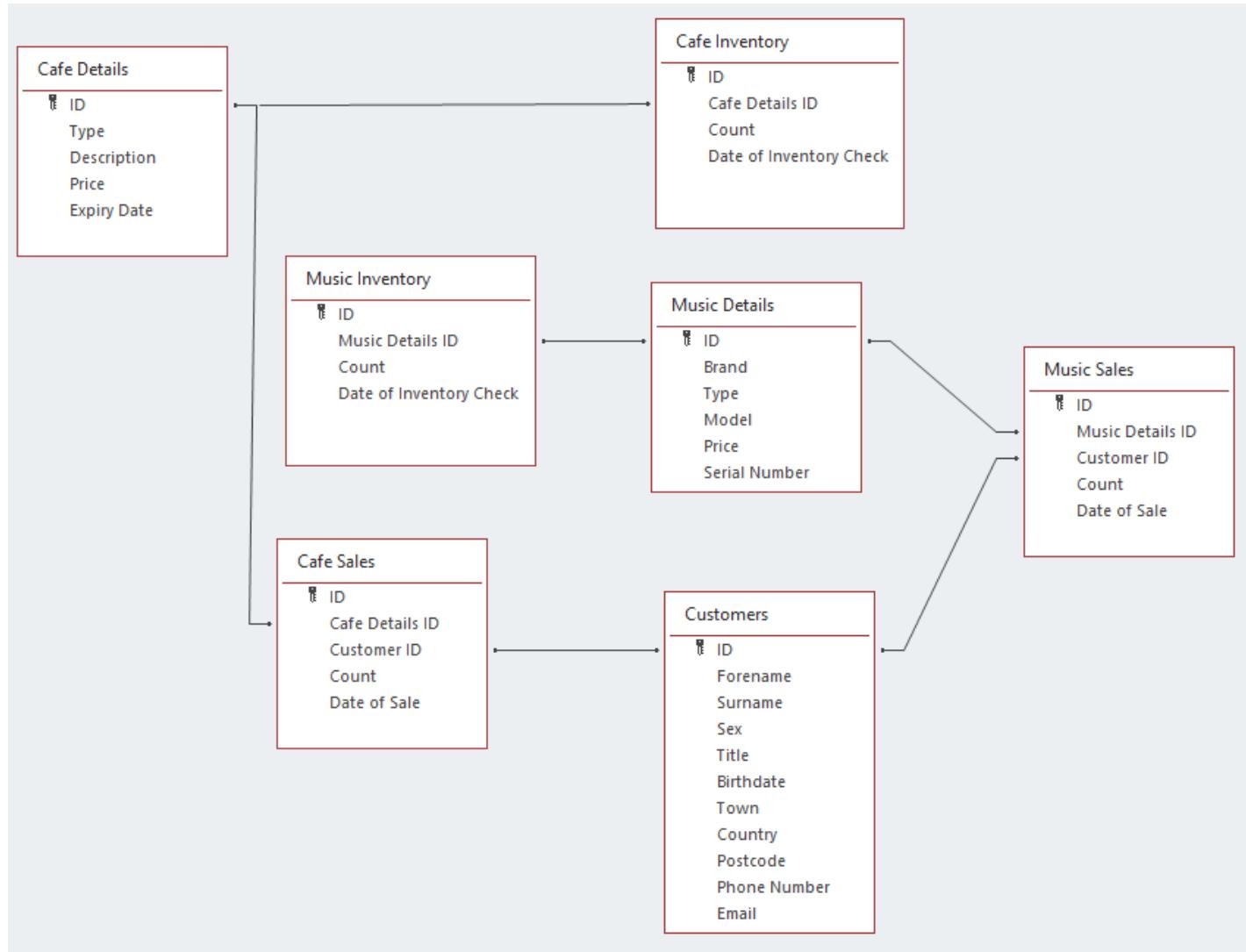
## Lesson Reports:

Each lesson report must have one student, one teacher, and one lesson plan linked.



**Inventory and Sales:**

Inventory and sales are recorded with links to sock details. Sales can also optionally have a linked customer ID.

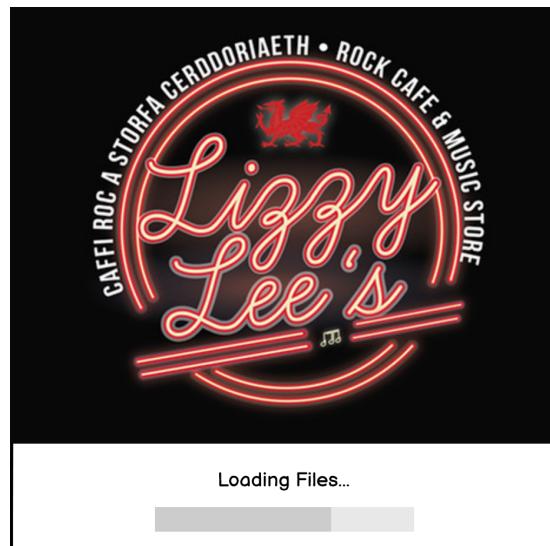


# Application Interface

This section will demonstrate and explain the full design that will be followed when programming the prototype to produce a software matching the specification demanded by the stakeholders of the current system from the Investigation document.

## SPLASH SCREEN

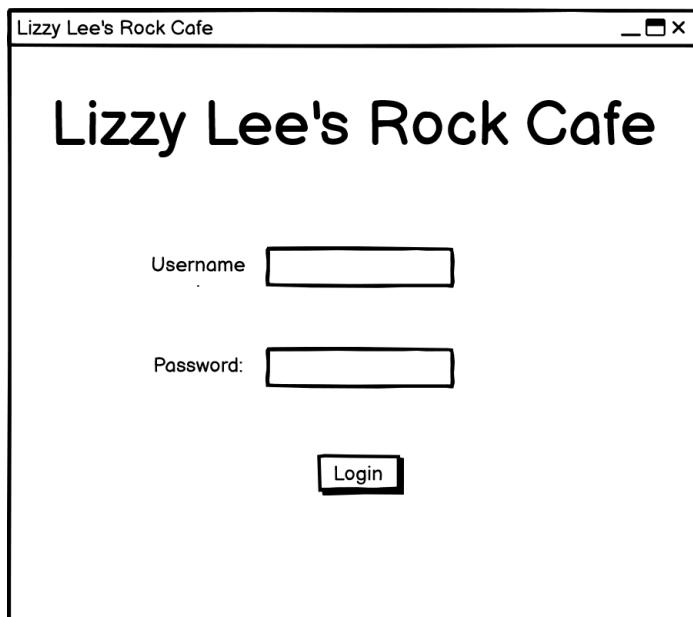
The document will launch with a splash screen displaying the Lizzy Lee's company logo. This decision was made not because of design choice but out of necessity due to the fact I will use asymmetric encryption to store and retrieve all data securely which can take some time for larger files to decrypt.



The loading bar will indicate to users that the app hasn't crashed if the loading takes longer than the user is expecting

## LOGIN SCREEN

All users will be greeted with a login screen after the program loads. Here they can enter their credentials and a match will send them to the corresponding landing screen depending on their role in the



The Username field will show the user's input

The Password field will hide the characters being typed to protect the user's data.



If the username and password fields don't match any stored credentials inform the user with a system prompt.



If no credentials are found in the system's database it will prompt the user to login with admin details.

# MAIN SCREEN

The main screen will be different according to which user is logged into the system, but for each user they will be able to find every feature they require from this screen. Managers will have access to every feature - so I will be using the manager login to demonstrate how each feature will look and be interfaced with in the final product - however I will also demonstrate examples of how a different user may have restricted access to some features.

## User Records Section:

ID	Name	Job Title	Title
1	Giacomo Guizzoni	Founder & CEO	Mr
2	Marco Botton	Tutuofare	Dr
3	Mariah MacLachlan	Better Half	Mrs
4	Valerie Liberty	Head Chef	Mr

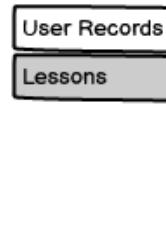
On the left is the section selection tab:

**Manager:**



Here users can choose to view any section of the system available to them. These tabs will swap the centre panel with the contents of the selected tab. The image on the left shows how the tab looks for managers, and the image on the right is how it will look for teachers.

**Teacher:**



In the centre panel a treeview table will display the table data for that section. Users can change the displayed data from the tabs available to them at the top.

Employee \ Teacher \ Student \ Customer					
ID	Name	Job Title	Title	...	...
1	Giacomo Guilizzoni	Founder & CEO	Mr	...	...
2	Marco Botton	Tuttofare	Dr	...	...
3	Mariah MacLachlan	Better Half	Mrs	...	...
4	Valerie Liberty	Head Chef	Mr	...	...

ID	<input type="text"/>	Title	<input type="text"/> Mr	<input type="button"/>	<input type="text"/>
Name	<input type="text"/>		<input type="text"/> Mrs	<input type="button"/>	<input type="text"/>
Job Title	<input type="text"/>		<input type="text"/> Dr	<input type="button"/>	<input type="text"/>
			<input type="text"/> ...	<input type="button"/>	<input type="text"/>

Search    Update Record    Add Record    Delete Record

Entry fields accepting user input for each column will be displayed near the bottom of this section. Users can select a record from the treeview, and its contents will be displayed in these entries. Users will then be able to edit the contents (with the correct permissions) and update the record, or delete the record by using the buttons displayed in the control section at the bottom.

Manager:

<input type="button"/> Search	<input type="button"/> Update Record	<input type="button"/> Add Record	<input type="button"/> Delete Record
-------------------------------	--------------------------------------	-----------------------------------	--------------------------------------

Depending on the users access rights they will be provided different options in this control section.

Teacher:

Update Record
---------------

*Teachers can only view their own record in the teachers tab  
so they should only be able to update it*

## Lessons Section:

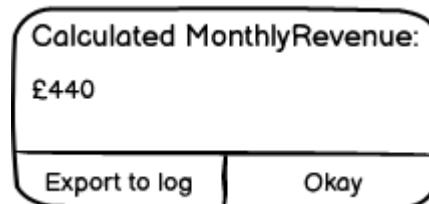
Teachers, students, and managers will be able to view the 'Lessons' section.

Lesson Bookings tab

ID	Student ID	TeacherID	Price	
1	10020211	01082439	65	
2	8763132	01082439	65	
3	10020217	00393102	55	

This section has a similar layout to 'User Records' however there is an additional button available to managers to display an estimated revenue from the selected records (or all if none are selected)

Clicking the button will display this pop-up:



The 'Lesson Plans' tab within the lessons section will also display the large text field of the 'Procedure' in a separate frame besides the treeview frame to the right.

### ‘Lesson Reports’ tab:

## Stock Section:

Users can access the stock section to make changes to inventory levels. The 'Cafe Inventory' and 'Music Inventory' tabs have buttons that will estimate the value of the current inventory.

'Cafe Inventory' tab:

ID	ItemID	Count	Date Added
1	10020211	12	...
2	8763132	2	...
3	10020217	302	...

ID:  Location: Room 12

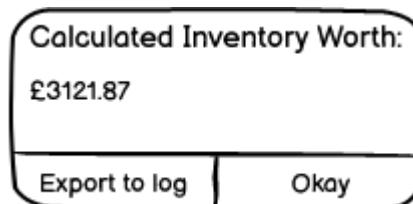
ItemID:  Date:

Count:

## 'Music Inventory' tab:

ID	ItemID	Count	Date Added
1	10020211	12	...
2	8763132	2	...
3	10020217	302	...

The 'Calculate Inventory' button will summon this pop-up to display the results of the calculation.



### 'Cafe Details' tab:

### ‘Music Details’ tab:

## Sales Section:

The 'Sales' section allows users to record item sales.

'Cafe Sales' tab:

ID	ItemID	CustomerID	Count	Date of Sale
1	10020211	0012910121	1	04/01/2023
2	8763132	2	...	
3	10020217	302	...	

A 'CustomerID' is optionally available. With this feature future development can lead to the system supporting customers to login and view a history of their sales, or additionally the company may wish to implement a 'loyalty scheme' using information from customers' sales.

The 'CustomerID' can still be used by staff to search for customer sales by searching the database using that user's ID as a filter.

'Music Sales' tab:

ID	ItemID	CustomerID	Count	Date of Sale
1	10020211	0012910121	1	04/01/2023
2	8763132	2	...	
3	10020217	302	...	

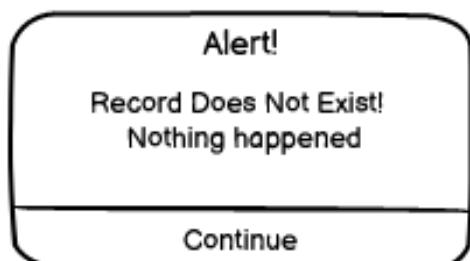
ID:   
 Date:   
 ItemID:   
 Count:

## System Alerts:

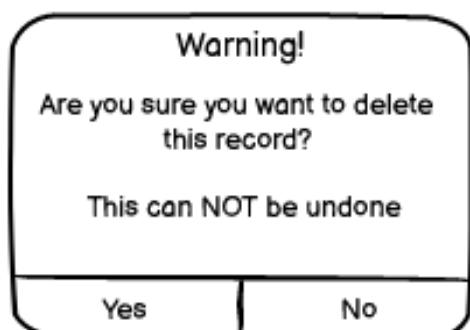
Entry input data will be validated. If a user tries to enter incorrectly formatted text into a field the system will warn the user:



If the input does not match the regular expression for each field when the user tries to add/ update a record the system will throw this error.



If the user tries to update or delete a record that does not exist this error will warn the user.



If the user chooses to delete a record there will be a warning to make sure the user is sure they want to delete the record.

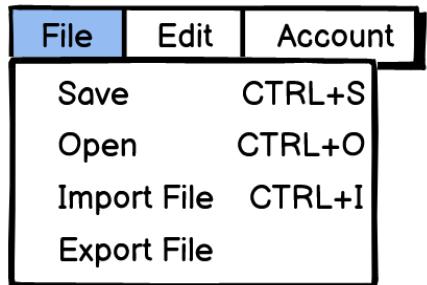
Additionally, any entries validated with a lookup check will simply display a dropdown of the available options instead of accepting user typed input.

**Title:**

A dropdown menu interface. The visible part shows the options 'Mr', 'Mrs', 'Ms', 'Dr', and '...', with a small downward arrow indicating more options are available. The input field contains the letter 'i'.

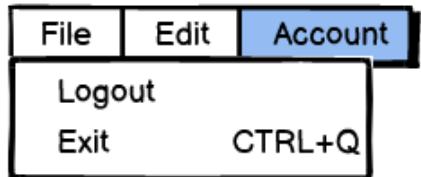
**Toolbar:**

At the top of the application window there will be a toolbar for users to execute certain system actions.



The 'File' menu will allow users to save their changes, import or export files, or edit their preferences.

They also have the option to open a backup file.



From the 'Account' menu users will be able to logout of the system or simply exit the app.

The system shortcuts are also displayed next to the options where a shortcut is allocated.

## Sorting:

Users will be able to sort the treeview results by clicking on any column header.

There will be three sorting types; Date sorting, Number sorting, and String sorting.

**Date sorting** will result in the records being sorted according to year, month, and date:

	Unsorted	Ascending	Descending
Hire Date	21/09/1998 08/12/1992 14/02/2000 07/09/2000	07/09/2000 14/02/2000 21/09/1998 08/12/1992	08/12/1992 21/09/1998 14/02/2000 07/09/2000

**Number sorting** will sort according to the value of the number. It will also apply to prices:

	ID	Ascending	Descending
ID	4 2 1 3	4 3 2 1	1 2 3 4

**String sorting** will sort according to the alphanumerical order:

Name
William Gill
Liz Pryde
Pat Lee

Name
Liz Pryde
Pat Lee
William Gill

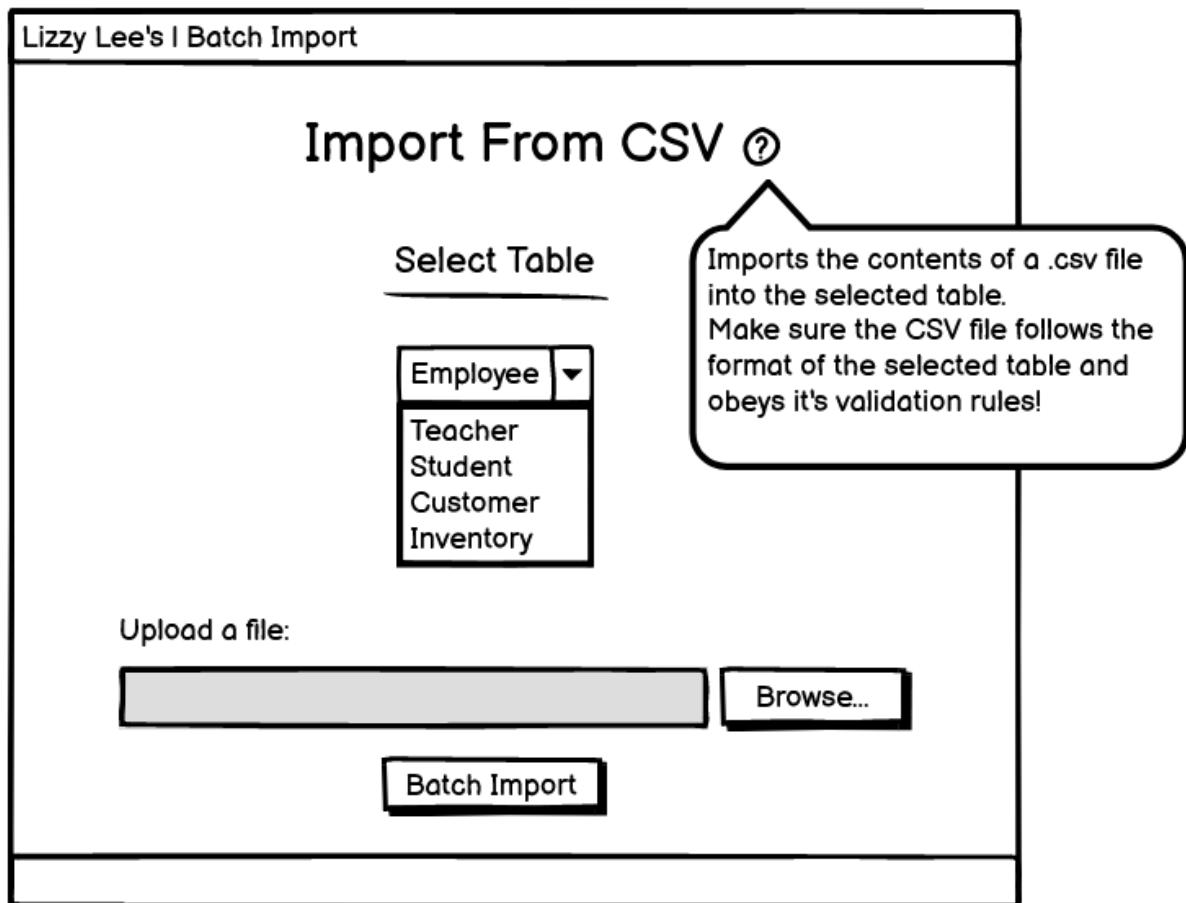
Name
William Gill
Pat Lee
Liz Pryde

For readability purposes only the *column being sorted by* is shown in the demonstrations above, however in the final application every column will be displayed when sorting.

## IMPORT SCREEN

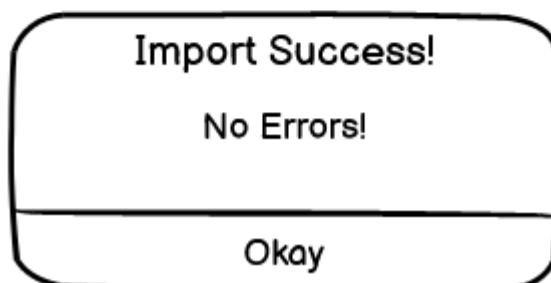
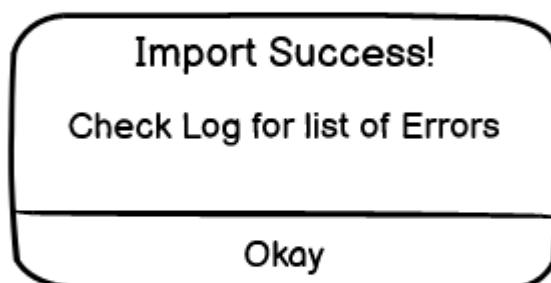
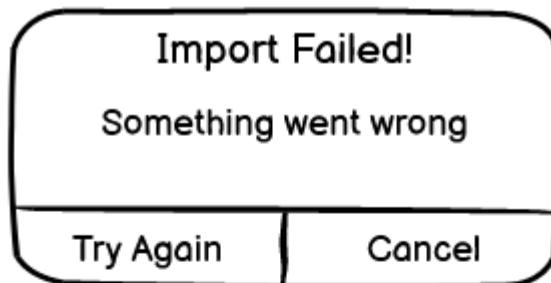
If a user with access rights needs to batch import data they can use the 'Batch Import' window accessible under the 'File -> Import File' option in the toolbar.

It will open a window giving the user the ability to choose a table to append data to, and accept a '.csv' file from the users file system.



The window is simple and uncomplicated for a lightweight feel that matches the requirements of the system. Additionally there will be a help button to explain the purpose of the screen to the user.

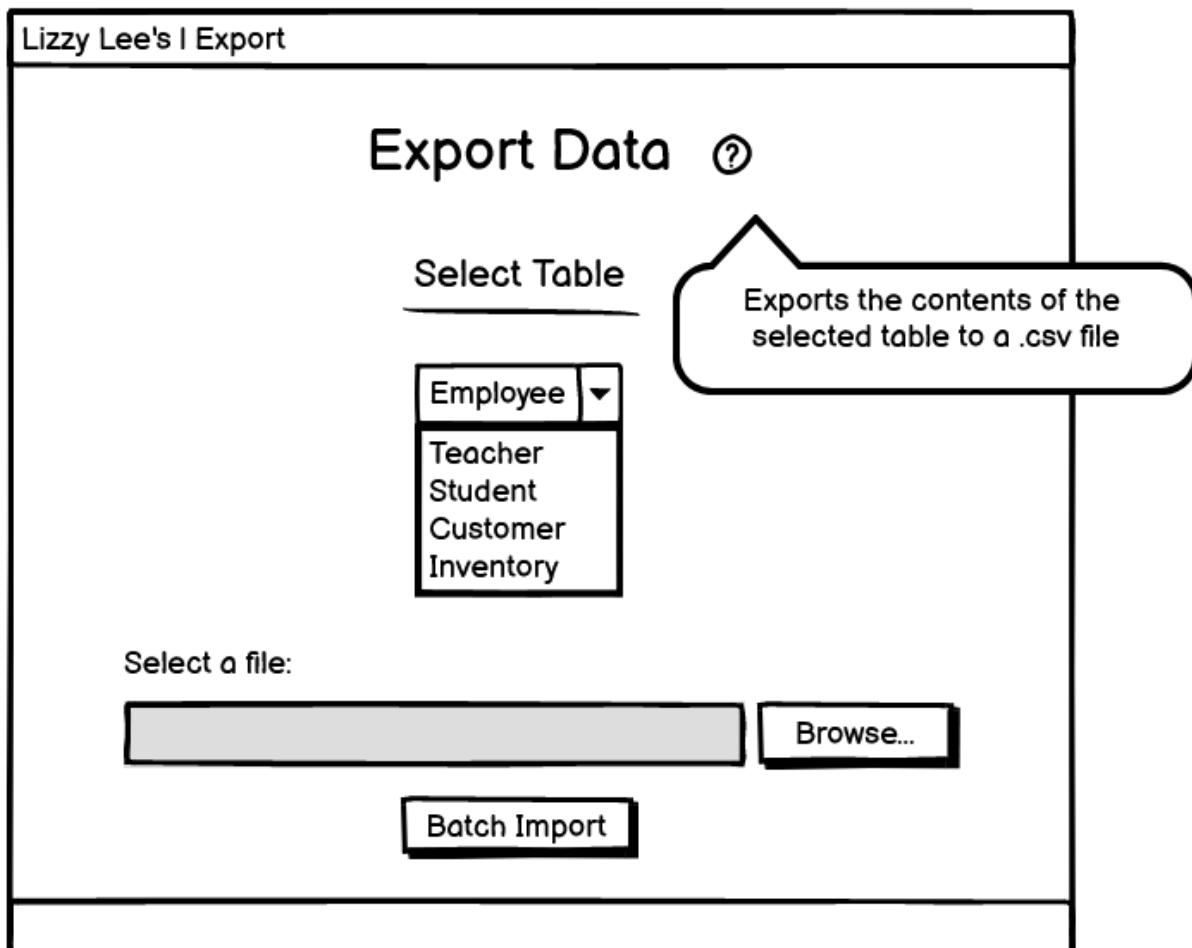
If an error is encountered during the importing sequence the system will display an error message:



If any errors are encountered they will be placed in a text log file where all system errors will be recorded.

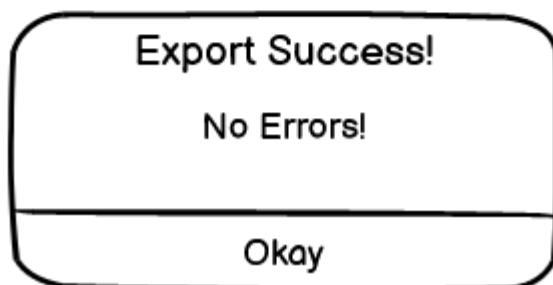
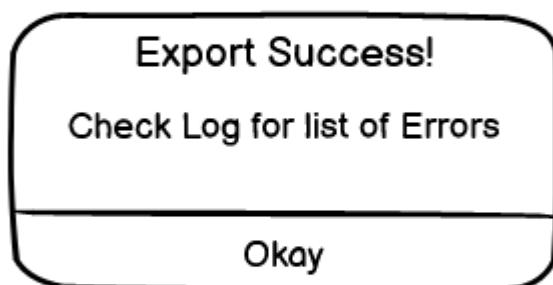
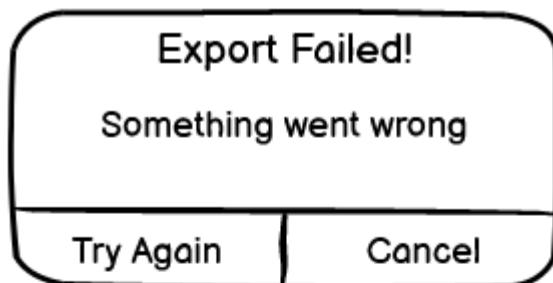
## EXPORT SCREEN

Users can also choose to export the entire contents of a table to a '.csv' file. Similar to the import screen the user can choose a table and destination file to export to by opening the 'Export' window under the 'File -> Export File' option in the toolbar.



The window is simple and uncomplicated for a lightweight feel that matches the requirements of the system. Additionally there will be a help button to explain the purpose of the screen to the user.

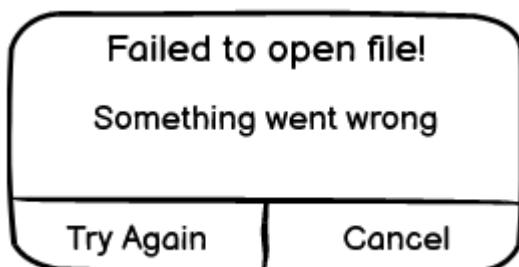
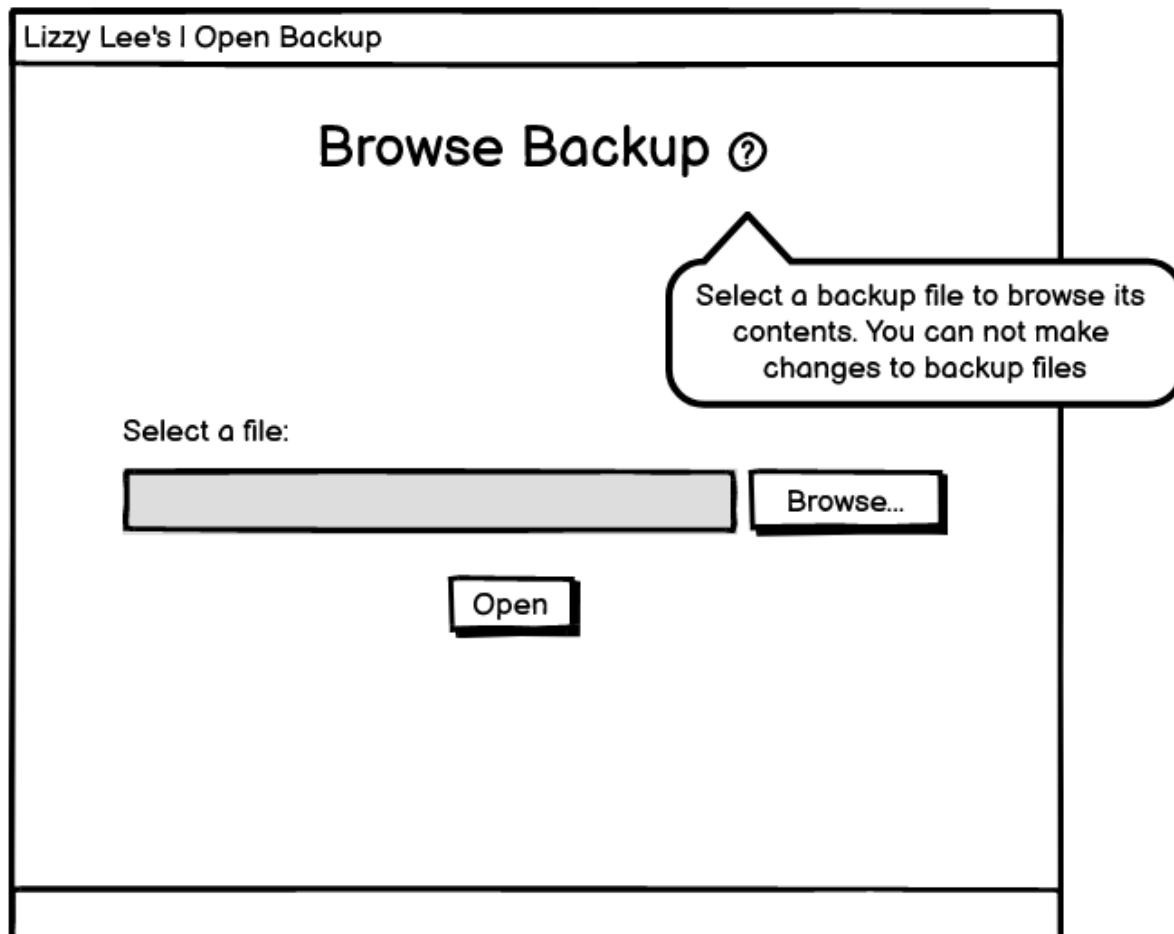
If an error is encountered during the importing sequence the system will display an error message:



If any errors are encountered they will be placed in a text log file where all system errors will be recorded.

## OPEN SCREEN

Managers can open a backup file to browse its data. If they desire to restore to this version they can rename the file to data.lzl and place it in the applications primary directory.



If the system fails to open the backup file this warning will alert the user.

# DATA

## SECURE STORAGE

All data will be stored in one permanent encrypted file when the system is not in use. To secure the data the system will use asymmetric encryption with a public and private key to store the data. When the program is running the data will be loaded into memory and committed back to a file only when the program is saved or exited.

### Storage Method

#### **Initialisation:**

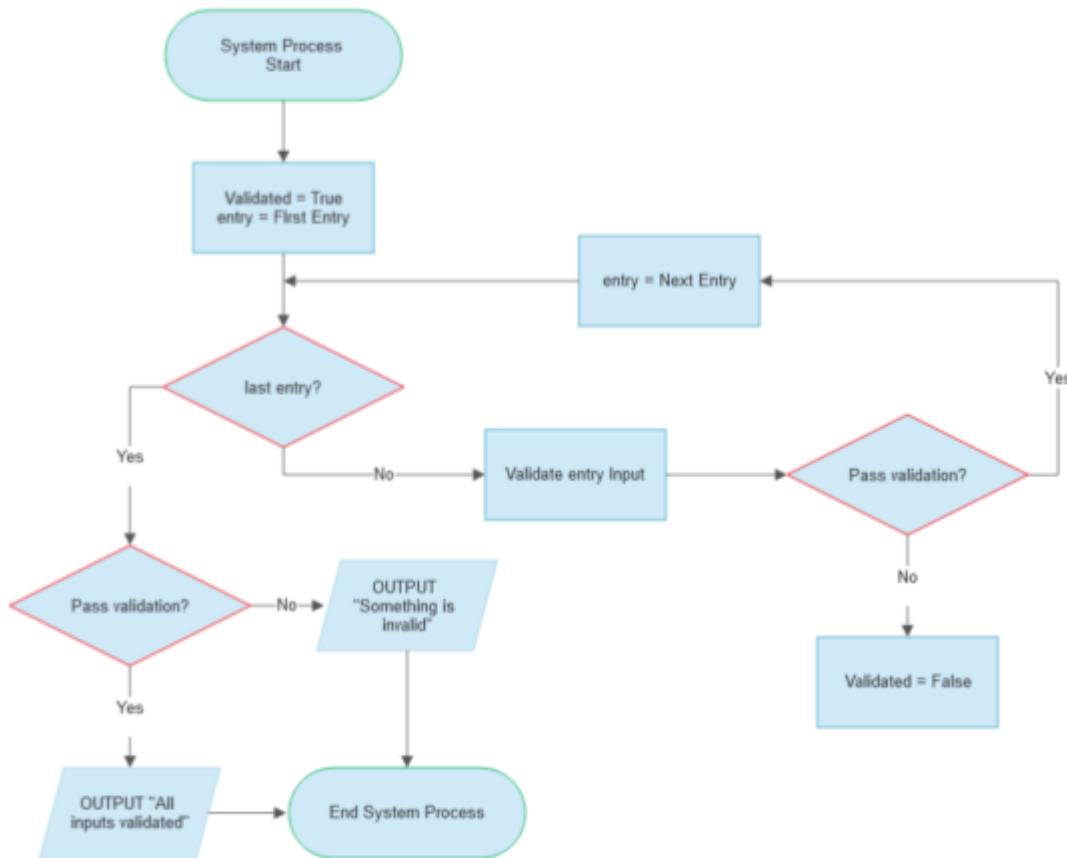
When the application starts if no data file exists then the system assumes it is the first time the program is being launched and create an empty sql database with the default hard-coded schema and the default admin user; Username:admin, Password:Passw0rd

#### **Execution/Saving:**

During the program's execution all data will be stored in memory in an unencrypted format. When the application is exited or the user manually saves, the program will securely save the program's data by using the 'SQLite3' module's 'iterdump' method to dump the sql schema and data from memory into a string which will then be encrypted using the public key, and stored into a .lzl file (custom file extension - essentially a .txt file).

#### **Application startup:**

Every subsequent time the program is executed it will take the private key hard coded into its source code (which will be compiled and unreadable) and decrypt the data file. The decrypted sql schema and data list will be executed by the program to rebuild the database in memory.



## Backups

### Frequency:

The system will backup data every other day and allows users to browse backup files without editing their contents. Backups will also be encrypted and can only be seen by a manager logging into the system and using the File -> Open option in the toolbar.

### Restoration:

Backups can be restored by simply copying the file and changing its name to data.lzl and placing it in the applications main directory. This is to ensure that data can be restored if the user's computer has lost its data and the only backup is on an external storage device.

### Method:

Backups will be made as the application is exited. The system will check the backup folder for the date of the last backup, and if the last backup was **more than 1 day ago** it will save a copy of the file as a backup in the folder with the date as the name.

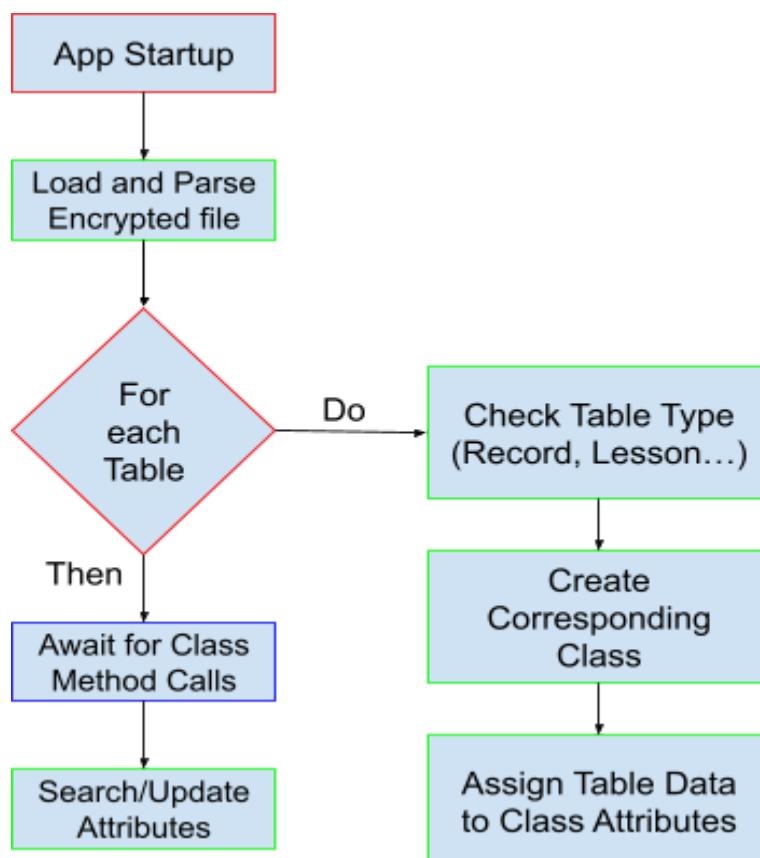
# VOLATILE DATA

The new system will store all runtime data in attributes within classes. Because the application will be created with a modular design, classes will create non-static variables and arrays will not be of predefined (hard-coded) size. Instead classes will read metadata from the SQL tables and create an array based on the table metadata.

This method will allow for the company to scale up their system in the future with ease as only the SQL data has to be edited.

Volatile data will be created with the following rules:

App Startup-> Get SQL tables -> read SQL table metadata -> create class for each table -> assign table data to class attributes.



With this any class can interact with other class methods to retrieve or edit data stored in that class attribute which relates to a certain SQL table. The classes also handle keeping their associated table up to date with data changes.

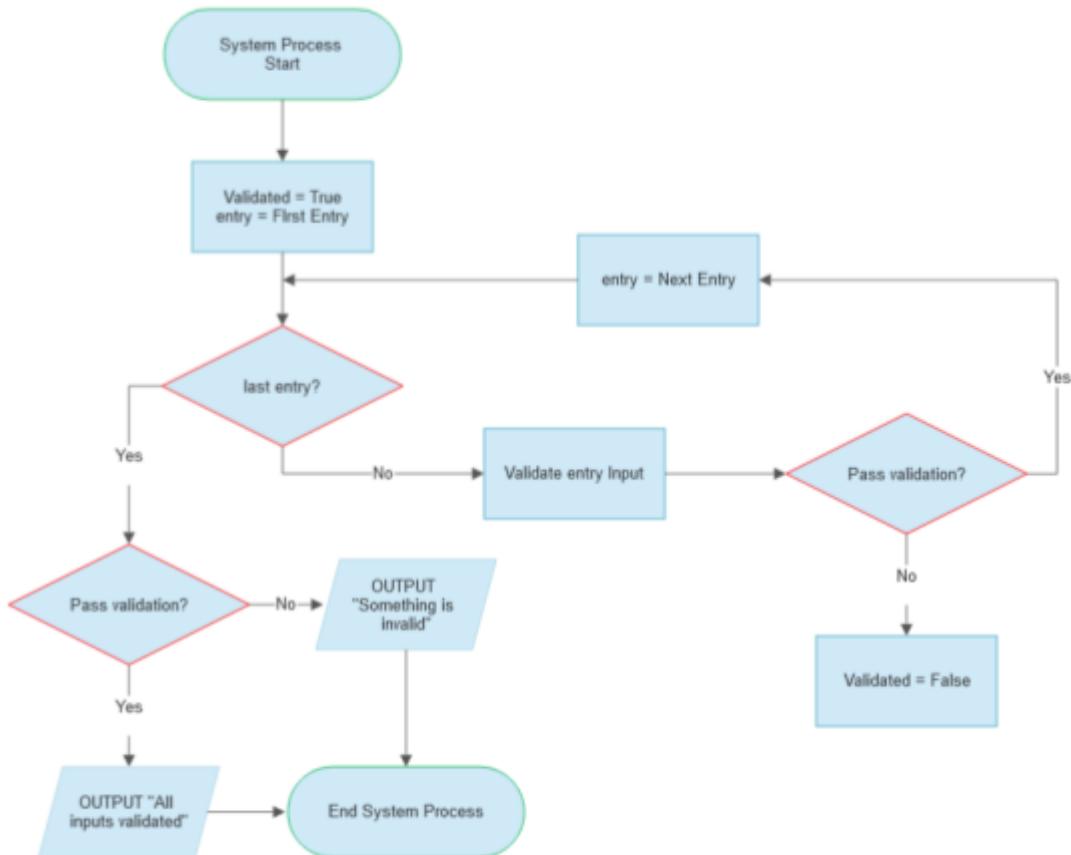
# SECURITY

## Validation

To ensure data integrity is upheld all data fields will be validated using python's regex library which has flexible use that can ensure data is validated and of the correct format. This will reduce the risk of incorrect data being input and stored in the system causing errors for the system and humans in the future.

### Validation system flowchart:

System services that utilise input data will follow a predefined system flow to ensure the data is validated. First the system will check each input field for applied validation rules, the system will then check the data against each of these rules for each input the function uses; if a single field fails the validation.



### Date validation:

Dates will follow the format DD/MM/YYYY

`re = "\d{2}/\d{2}/\d{4}$"`

- ‘^’ anchor marks the beginning of the string
- ‘\d’ means any digit (0-9)
- {2} means 2 of the previously stated character
- ‘/’ is simply the forward slash character
- ‘\$’ anchor marks the end of the string

**Email validation:**

Emails must have any number of characters, followed by an ‘@’, then any number of characters, followed by a ‘.’ and any number of characters.

**re = ‘^\S+@\S+\.\S+\$’**

- ‘^’ anchor marks the beginning of the string
- ‘\S+’ means any number of characters, uppercase or lowercase
- ‘@’ is the @ symbol
- ‘.’ marks a ‘.’
- ‘\$’ anchor marks the end of the string

**Username validation:**

Usernames must begin with either an uppercase or lowercase letter, followed by any number of characters.

**re = ‘^([A-Z]|[a-z])\S+\$’**

- ‘^’ anchor marks the beginning of the string
- ‘[A-Z] | [a-z]’ marks any alphanumeric character, uppercase or lowercase.
- ‘\S+’ means any number of characters, uppercase or lowercase
- ‘\$’ anchor marks the end of the string

**Password validation:**

Passwords must contain at least one uppercase, lowercase, number, and special character and be over 8 digits long.

**re = ‘^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@\$!%\*?&])[A-Za-z\d@\$!%\*?&]{8,}\$’**

- ‘^’ anchor marks the beginning of the string
- ‘(?=.\*[A-Z])’ ensures that there is at least one uppercase letter
- ‘(?=.\*[a-z])’ ensures that there is at least one lowercase letter
- ‘(?=.\*[@\$!%\*?&])’ ensures that there is at least one special character
- ‘(?=.\*\d)’ ensures that there is at least one digit
- ‘[A-Za-z\d@\$!%\*?&]{8,}’ matches any character eight times or more, ensuring that the password has at least 8 characters in total

**Name validation:**

Names must begin with either an uppercase letter, followed by any number of lowercase letters.

**re = ‘^ [A-Z] [a-z]+ \$’**

- ‘^’ anchor marks the beginning of the string
- ‘[A-Z]’ marks any alphanumeric character, uppercase or lowercase.
- ‘[a-z]+’ marks any number of lowercase letters
- ‘\$’ anchor marks the end of the string

**Phone number validation:**

Phone numbers must match the UK phone number format, starting with either a +44 or a 0, and ending in 10 digits.

**re = ‘^ \+44\d{10} \$ | ^ (0\d{10}) \$’**

- ‘^’ anchor marks the beginning of the string
- ‘\+’ marks the + symbol
- ‘\d{10}’ marks 10 digits from 0-9
- ‘\$’ anchor marks the end of the string
- ‘|’ means matching either the expression before or after

# DATA TABLES

The system will use Python's SQLite3 module. Following are each of the SQL tables that will be loaded into memory during program execution, and stored in permanent storage outside of program use.

## **Customers Table**

This table stores essential data for customers.

Field Name	Data Type	Field length	Validation type	Overview of validation rule applied	Description of field	Typical Data
ID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique value calculated from highest existing number in customer file	80097623
username	String	24	Format Check	Can not start with number	User's unique username used to login to the system	sirJohnWilliams1989

password	String	24	Format Check	Can not start with number, must include uppercase, lowercase, symbol, and number	User's unique hidden password used to login to the system	Passw0rd!
forename	str	24	Type check	Characters Must only be in range A-Z	Details the Customer's forename	John
surname	str	24	Type check	Characters Must only be in range A-Z	Details the Customer's surname	Smith
sex	Bool	1	Lookup Check	Must Be in selection ["Male", "Female"]	Field that indicates the customer's Sex	M
title	String	3	Lookup Check	Must Be in selection ["Mr", "Mrs" "Ms", "Dr"]	Field that indicates the customer's title for addressing	Mr
birthdate	str	10	Format check, Range check	Must be valid Date, must be in the past	Details the Customer's birthdate	01/03/1998
town	str	30	Type check	Characters Must only be in range A-Z	Details the Customer's town	Llangefni
county	str	15	Lookup check or Type check	Characters Must only be in range	Details the Customer's given	Anglesey

				A-Z	county	
postcode	str	8	Format check - must be "LL00 0LL"	The Input must be a valid postcode format	Details the Customer's postcode	LL567HY
phone_num	str	11 (How many digits in phone number are there)	Length check	Phone Number must be 11 digits longs as that is the length of a UK phone number	Details the Customer's phone number for contact	"07726273604"
email	str	40	Format includes @ and . or Presence check	Input must not be blank, and there must be an '@' symbol to ensure it is an email that was inputted	Details the Customer's email for contact	johnsmith@mail.co m

### Employees Table

This table holds the login data for employees and their personal and contact details.

Field Name	Data Type	Field length	Validation type	Overview of validation rule applied	Description of field	Typical Data
ID	String	8	Format Check	Must be unique "99999999", "99999998"	Unique value calculated from highest existing number in Employee	80097623

					file	
username	String	24	Format Check	Can not start with number	User's unique username used to login to the system	EdwardsJrCat
password	String	24	Format Check	Can not start with number, must include uppercase, lowercase, symbol, and number	User's unique hidden password used to login to the system	Passw0rd!
forename	str	24	Type check	Characters Must only be in range A-Z	Details the Employee's forename	John
surname	str	24	Type check	Characters Must only be in range A-Z	Details the Employee's surname	Smith
sex	Bool	1	Lookup Check	Must Be in selection ["Male", "Female"]	Field that indicates the Employee's Sex	M
title	String	3	Lookup Check	Must Be in selection ["Mr", "Mrs", "Ms", "Dr"]	Field that indicates the Employee's title for addressing	Mr
birthdate	str	10	Format check, Range check	Must be valid Date, must be in the past	Details the Employee's birthdate	01/03/1998

hire_date	str	10	Format check, Range check	Must be valid Date, must be in the past	Details the Employee's date of hire	01/03/2009
job_description	str	20	Lookup check	Must be from provided selection: [Manager, Crew]	Details the Employee's job role	Manager
town	str	30	Type check	Characters Must only be in range A-Z	Details the Employee's town	Llangefni
county	str	15	Lookup check or Type check	Characters Must only be in range A-Z	Details the Employee's given county	Anglesey
postcode	str	8	Format check - must be "LL00 0LL"	The Input must be a valid postcode format	Details the Employee's postcode	LL567HY
phone_num	str	11 (How many digits in phone number are there)	Length check	Phone Number must be 11 digits long as that is the length of a UK phone number	Details the Employee's phone number for contact	"07726273604"
email	str	40	Format includes @ and . or Presence check	Input must not be blank, and there must be an '@' symbol to ensure it is an email that was inputted	Details the Employee's email for contact	johnsmith@mail.com

**Teachers Table**

This table holds the login data for teachers and their personal details.

Field Name	Data Type	Field length	Validation type	Overview of validation rule applied	Description of field	Typical Data
ID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique value calculated from highest existing number in Teacher file	80097623
username	String	24	Format Check	Can not start with number	User's unique username used to login to the system	Jackellandhyde
password	String	24	Format Check	Can not start with number, must include uppercase, lowercase, symbol, and number	User's unique hidden password used to login to the system	Passw0rd!
forename	str	24	Type check	Characters Must only be in range A-Z	Details the Teacher's forename	John

surname	str	24	Type check	Characters Must only be in range A-Z	Details the Teacher's surname	Smith
sex	Bool	1	Lookup Check	Must Be in selection ["Male", "Female"]	Field that indicates the Teacher's Sex	M
title	String	3	Lookup Check	Must Be in selection ["Mr", "Mrs" "Ms", "Dr"]	Field that indicates the Teacher's title for addressing	Mr
birthdate	str	10	Format check, Range check	Must be valid Date, must be in the past	Details the Teacher's birthdate	01/03/1998
hire_date	str	10	Format check, Range check	Must be valid Date, must be in the past	Details the Teacher's date of hire	01/03/2009
town	str	30	Type check	Characters Must only be in range A-Z	Details the Teacher's town	Llangefni
county	str	15	Lookup check or Type check	Characters Must only be in range A-Z	Details the Teacher's given county	Anglesey
postcode	str	8	Format check - must be "LL00 0LL"	The Input must be a valid postcode format	Details the Teacher's postcode	LL567HY

phone_num	str	11 (How many digits in phone number are there)	Length check	Phone Number must be 11 digits long as that is the length of a UK phone number	Details the Teacher's phone number for contact	"07726273604"
StudentIDs	String	8	Format Check	Must be unique "99999999", "99999998"	Unique value calculated from highest existing number in Teacher file	80097623

**Students Table**

This table holds the login data for students and their personal details as well as info on their music vocation.

Field Name	Data Type	Field length	Validation type	Overview of validation rule applied	Description of field	Typical Data
ID	String	8	Format Check	Must be unique "99999999", "99999998"	Unique identifier calculated from highest existing number in Student table	80097623
username	String	24	Format Check	Can not start with number	User's unique username used to login to the system	sirJohnWilliams1989

password	String	24	Format Check	Can not start with number, must include uppercase, lowercase, symbol, and number	User's unique hidden password used to login to the system	FrenchBull52190
forename	str	24	Type check	Characters Must only be in range A-Z	Details the Student's forename	John
surname	str	24	Type check	Characters Must only be in range A-Z	Details the Student's surname	Smith
sex	Bool	1	Lookup Check	Must Be in selection ["Male", "Female"]	Field that indicates the Student's Sex	M
title	String	3	Lookup Check	Must Be in selection ["Mr", "Mrs" "Ms", "Dr"]	Field that indicates the Student's title for addressing	Mr
birthdate	str	10	Format check, Range check	Must be valid Date, must be in the past	Details the Student's birthdate	01/03/1998
town	str	30	Type check	Characters Must only be in range A-Z	Details the Student's town	Llangefni
county	str	15	Lookup check or Type check	Characters Must only be in range A-Z	Details the Student's given county	Anglesey

postcode	str	8	Format check - must be "LL00 0LL"	The Input must be a valid postcode format	Details the Student's postcode	LL567HY
guardian_contact_phone_num	str	11 (How many digits in phone number are there)	Length check	Phone Number must be 11 digits longs as that is the length of a UK phone number	Details the Student's guardian's phone number for contact	"07726236403"
phone_num	"	"	"	"	Details the Student's phone number for contact	"
email	str	40	Format includes @ and . or Presence check	Input must not be blank, and there must be an '@' symbol to ensure it is an email that was inputted	Details the Student's email for contact	johnsmith@mail.co m
Instrument	str	30	Type check, Format check	Must only contain alphanumerical characters	Details the Student's chosen instrument	Acoustic Guitar
Skill Level	str	12	Lookup Check	Must be either: Beginner, Intermediate, Skilled, or Pro	Details the student's skill level with their chosen instrument	Intermediate

**Student Bookings Table**

This table stores bookings between students and teachers

Field Name	Data Type	Field length	Validation type	Overview of validation rule applied	Description of field	Typical Data
ID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique identifier calculated from highest existing number in customer table	80097623
StudentID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique identifier calculated from highest existing number in customer table	80097623
TeacherID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique identifier calculated from highest existing number in customer table	80097623
location	str	12	Lookup Check	Must be an item from a selection	Details where the lessons will take place	Basement
lesson_frequency	int	2	Lookup Check	Must be one of [1, 2, 4]	How many lessons the student has per month	2

lesson_day	str	12	Lookup Check	Must be in day that is provided [Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday]	What day the student has lessons	Thursday
lesson_time	str	5	Lookup Check	Must be in time slot provided for given day e.g [13:30, 14:00, 15:30, 16:00]	What time the student has lessons	16:30
lesson_length	int	2	Lookup Check	Must be given length [20, 30, 45, 60]	How long the lesson lasts (in minutes)	30
lesson_cost	float	4	Type check	Price Must be an float	Details the Customer's town	Llangefni
booking_date	str	10	Format check, Range check	Must be valid Date, must be in the past	Details the Student's birthdate	01/03/1998
booking_update	str	10	Format check, Range check	Must be valid Date, must be in the past	Details the Student's birthdate	01/03/1998
cancelled	bool	1	Type check	Must be either Yes or No	Whether or not the lesson has been cancelled	No

**Lesson Plans Table**

Stores information about lesson plans provided to teachers

Field Name	Data Type	Field length	Validation type	Overview of validation rule applied	Description of field	Typical Data
ID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique identifier calculated from highest existing number in LessonPlans table	10089112
Lesson Title	String	30	Type Check	Must be a string	Title of Lesson	Introduction to Guitar Basics
Lesson Objective	String	50	Type Check	Must be a string	Goal to be achieved by students in the lesson	Students will learn the basic concepts of playing guitar, including proper posture, hand positioning, and basic chords.
Materials Required	String	50	Type Check	Must be a string	Instruments and other items required for the lesson	Music sheet no.5, Capo, Guitar Pick
Procedure	String	50000	Type Check	Must be a string	Details the Student's surname	[LONG DESCRIPTION OF PROCEDURE OF

						LESSON]
--	--	--	--	--	--	---------

**Lesson Report Table:**

Table for resting lesson reports for students

ID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique identifier calculated from highest existing number in LessonPlans table	10089112
LessonPlansID	String	8	Format Check	Must be unique “99999999”, “99999998”	ID of the lesson plan	10089112
StudentID	String	8	Format Check	Must be unique “99999999”, “99999998”	ID of the student	10089112
TeacherID	String	8	Format Check	Must be unique “99999999”, “99999998”	Teacher's ID	10089112
Attended	Boolean	1	Type check	Must be Yes or No	Describes whether or not the student was present for the lesson	Yes/1

Student Behaviour	Int	2	Range Check	Must be in range 0-4	Describes the students behaviour during the lesson on a scale from 0-4, 4 being well mannered and 1 being unteachable. Additionally 0 if student wasn't present	4
Notes	String	50000	Type Check	Must be a string	Any additional notes the teacher wishes to record	Student was eager to learn ahead of assigned level
Date	Date	10	Type Check	Must be a Date	Date of the lesson	10/12/21

**Music Stock Table**

Stores details about stock items in the music store.

Field Name	Data Type	Field length	Validation type	Overview of validation rule applied	Description of field	Typical Data
Item ID	String	8	Format Check	Must be unique "99999999", "99999998"	Unique identifier calculated from highest existing number in MusicDetails table	10089112
Brand	String	50	Type Check	Must be a string	Name of item Brand	Gear4Music
Type	String	50	Lookup Check	Must be one of a selection: [Instrument, Cable...]	What type of item is being stored	Amp
Model	String	50	Type Check	Must be a string	Item Model	Roland Cube 80X
Price	Float	6	Format Check	Must be in format £00.00	Price of Item	213.45
Serial Number	String	16	Format Check	Must be unique "99999999", "99999998"	Unique Serial Number on Item	AX011892E

**Cafe Stock Table:**

Stores details about stock items in the cafe store.

Field Name	Data Type	Field length	Validation type	Overview of validation rule applied	Description of field	Typical Data
Item ID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique identifier calculated from highest existing number in CafeDetails table	10089112
Type	String	50	Lookup Check	Must be one of a selection: [Coffee, Pastry...]	What type of item is being stored	Dessert
Description	String	50	Type Check	Must be a string	Description of the item	Brownie
Price	Float	6	Format Check	Must be in format £00.00	Price of Item	0.85
Expiry Date	Date	10	Type Check	Must be a Date	Date Item will expire	10/12/23

**Cafe Inventory Table:**

Table for recording inventory levels of the cafe at a specific date.

ID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique identifier calculated from highest existing number in MusicDetails table	10089112
CafeDetails ID	String	8	Format Check	Must be unique “99999999”, “99999998”	ID of the cafe item	10089112
Count	Int	4	Range check	Must be a between 0 and 1000	Number of this item held	1
Date of Inventory Check	Date	10	Type Check	Must be a Date	Date Item was recorded in inventory	10/12/21

**Music Store Inventory Table:**

Table for recording inventory levels of the music store at a specific date.

ID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique identifier calculated from highest existing number in MusicDetails table	10089112
MusicDetails ID	String	8	Format Check	Must be unique “99999999”, “99999998”	ID of the music item	10089112
Count	Int	4	Range check	Must be a between 0 and 1000	Number of this item held	1
Date of Inventory Check	Date	10	Type Check	Must be a Date	Date Item was recorded in inventory	10/12/21

**Cafe Sales Table:**

Records sales of items in the cafe, and can record the buyers customer ID if it exists.

ID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique identifier calculated from highest existing number in CafeSales table	10089112
CafeDetails ID	String	8	Format Check	Must be unique “99999999”, “99999998”	ID of the cafe item	10089112
Customer ID	String	8	Format Check	Must be unique “99999999”, “99999998”	ID of the customer	10089112
Count	Int	4	Range check	Must be a between 0 and 1000	Number of this item sold	1
Date of Sale	Date	10	Type Check	Must be a Date	Date Item was sold	10/12/21

**Music Store Sales Table:**

Records sales of items in the music store, and can record the buyers customer ID if it exists.

ID	String	8	Format Check	Must be unique “99999999”, “99999998”	Unique identifier calculated from highest existing number in MusicSales table	10089112
MusicDetails ID	String	8	Format Check	Must be unique “99999999”, “99999998”	ID of the Music item	10089112
Customer ID	String	8	Format Check	Must be unique “99999999”, “99999998”	ID of the customer	10089112
Count	Int	4	Range check	Must be a between 0 and 1000	Number of this item sold	1
Date of Sale	Date	10	Type Check	Must be a Date	Date Item was sold	10/12/21

# CLASSES

The program will make use of object oriented programming to reduce code lines and ensure the program is dry.

Classes will also drastically reduce the number of globals required by the program. Without classes a Tkinter based application requires the use of globals to pass data between windows, however with classes most variables can be inherited and passed by parameters drastically reducing the number of globals and freeing up memory.

## File Handler Class

This class will handle the opening and saving plus the encryption and decryption of the system's files.

### Methods

Name	Function
__init__	Initialises the Application with default values
openFile	Opens a file
closeFile	Saves the file
encrypt	Encrypts and returns input string
decrypt	Decrypts and returns input string

## Application Class(tkinter Tk)

The application class will be used as a superclass for creating more applied screens.

### Methods

Name	Function
__init__	Initialises the Application with default values

## Splash Screen Class(Application)

This subset class will initialise a splash screen until it is closed externally using its own methods.

### Methods

Name	Function
<code>__init__</code>	Initialises the Application with default values

## Login Class(Application)

This subset class will create a login window and handle login functions.

### Methods

Name	Function
<code>__init__</code>	Initialises the Application with default values
<code>skip</code>	<b>Dev method for debugging.</b> Allows the developer to skip the login and gain manager access
<code>checkCredentials</code>	Checks the users credentials against the database and responds according to the match

## Primary Application Class(Application)

This subset class controls the main application loop.

### Methods

Name	Function
<code>__init__</code>	Initialises the Application with default

	values
onExit	Method to call when the app is closed or exited
importWin	Opens an import window
importCSV	Takes input csv file and imports data to selected table
exportWin	Opens the export window
exportTable	Exports the selected table to a file

This class will hold the majority of variables that will be needed throughout the program's execution.

### **Primary Tab Class(tkinter Frame)**

This class will create a tkinter Frame for organising nested frames in a tab system.

Methods

Name	Function
__init__	Initialises the class with default variables

### **Primary Tab Class(tkinter Frame)**

This class will create a tkinter Frame for organising nested frames in a tab system.

Methods

Name	Function
__init__	Initialises the class with default variables

### **Table Frame Class(tkinter Frame)**

This class will create a tkinter Frame for organising nested frames in a tab system.

Methods

Name	Function
__init__	Initialises the class with default variables
fillTable	Populates the treeview list with sql results
headingPushed	Callback method for when a column header is pushed
sortDate	Sorts a list of dates
sortString	Sorts a list of strings
sortNum	Sorts a list of numbers

## Text Frame Class(tkinter Frame)

This class will create a tkinter Frame for displaying scrollable text

### Methods

Name	Function
__init__	Initialises the class with default variables

## Record Entry Class(tkinter Entry)

Class for record entries to validate themselves according to their column type when a user inputs data

Name	Function
__init__	Initialises the class with default variables
validate	Calls a validation method depending on the entry column type
validateID	Validates according to ID regular expression rules
validateDate	Validates according to date regular expression rules

validateEmail	Validates according to email regular expression rules
validateUsername	Validates according to username regular expression rules
validateName	Validates according to naming convention regular expression rules
validatePhone	Validates according to UK phone number regular expression rules

## Record Class(tkinter Frame)

Class for creating a dynamic frame to display a record's SQLite table contents and be editable.

Name	Function
__init__	Initialises the class with default variables
selectedItem	Gets information about selected treeview item
addRecord	Adds a record to the sql table
updateRecord	Updates a record in the sql table
validate	Loops through each RecordEntry and checks they are all valid with their own methods
deleteRecord	Deletes a record in the sql table
clearFields	Clears all the RecordEntry fields
searchQuery	Executes an sql statement to search the table for values in the RecordEntry widgets

## Booking Class(Record)

Subclass of record for creating a dynamic frame to display a booking's SQLite table contents and be editable.

Name	Function

__init__	Initialises the class with default variables
calculateCost	Calculates an estimate cost for selected bookings in a month
exportCalculation	Exports result of calculation to log file

# Pseudocode Algorithms

Each function the program requires will make use of many algorithms to achieve complicated tasks. Here I will create pseudo-code for each algorithm that may be used multiple times throughout the entire software. Because of this I will ensure the algorithms are efficient and versatile to reduce limitations by the system.

## DATA ENCRYPTION

To secure the information the new system will handle we will be storing the database files in an encrypted file using asymmetric encryption. This is a very secure type of encryption that can only be cracked by obtaining both the public and private key - however it is also very complicated and can be intensive for the system to decrypt large amounts of data. Because of this the system will decrypt the entire file system at startup with a splash screen, and store the data in memory. With this method the application will run smoothly without having to wait for each data query to be decrypted by instead accessing the fully decrypted database stored in memory.

Asymmetric encryption works by using complicated math made public by the creators of the RSA algorithm. This information is publicly available online and can be used to create functions that use public keys to encrypt data, and private keys to decrypt data:

**Public key:** Two numbers: a **product of two large prime numbers**, and another **random number**.

**Private key:** A **private number** calculated with the inverse modulo of the **two large prime numbers** and the **random number** in the public key.

### Calculate Mod Inverse:

To calculate the private key an algorithm is required to calculate the inverse mod of the two large prime numbers and a random number. Mod inverse is a complicated function that requires a method that can find the greatest common divisor of two numbers by recursively calling itself.

```
FUNCTION euclideanGreatestCommonDivisor(a, b) :  
  
    # Check if the first integer is 0  
    IF a == 0 THEN  
        RETURN (b, 0, 1)
```

```

    ELSE
        # Recursively calculate the greatest common divisor
        commonDivisor, y, x = euclideanGreatestCommonDivisor(b
        % a, a)
        RETURN (commonDivisor, x - (b // a) * y, y)
    END IF

END FUNCTION

FUNCTION modinv(a, modulo):
    g, x, y = euclideanGreatestCommonDivisor(a, modulo)

    IF g != 1 THEN
        RAISE Exception('modular inverse does not exist')

    ELSE:
        RETURN x % modulo
    END IF

END FUNCTION

```

## **Encrypt:**

This algorithm will take an input message and public key and return the encrypted message as a string using asymmetric encryption.

The algorithm will use the following standard functions that will remain abbreviated in the pseudocode to improve readability:

### **chr(ASCII):**

This function takes an ASCII code as a parameter and returns the character relating to that code.

### **pow(num, power):**

This function returns the mathematical result of the first parameter being raised to the power of the second parameter.

### **%:**

This operator is the mathematical operation modulo which returns the remainder of the division of the number before the operator, by the number after the operator.

```

FUNCTION encrypt(message, publicKey):
    # Empty string for encrypted message
    encryptedMsg = EMPTY STRING

    # For each character in the message
    FOR EACH char IN message DO
        # Get the character code of the character
        charCode = ASCII value of char

        # Encrypt the code using the public key
        encryptedChar = chr(pow(charCode, value of publicKey[1]) %
value of publicKey[0])

        # Append the encrypted character to the encrypted message
        encryptedMsg += encryptedChar
    END FOR

    # Return the encrypted message
    RETURN encryptedMsg

END FUNCTION

```

## Decrypt:

Uses the same functions as encrypt and uses the private key to decrypt the input string and returns the decrypted output.

```

FUNCTION decrypt(encryptedMessage, privateKey):
    decryptedMessage = EMPTY STRING
    # For each character in the encrypted message
    FOR EACH char IN encryptedMessage DO
        # Decrypt the character code using the private key
        decryptedChar = chr(pow(ord(char), value of privateKey[1]) %
value of privateKey[0])

```

```
# Append the decrypted character to the decrypted message
decryptedMessage += decryptedChar
END FOR

# Return the decrypted message
RETURN decryptedMessage

END FUNCTION
```

# DATA MANAGEMENT

I will make use of multiple sorting algorithms to make the user's record management tasks simpler. Each sorting algorithm will use the 'merge sort' technique justified by the fact it is relatively simple to program and is also one of the most efficient sorting algorithms due to its use of recursion.

## File Access:

The program will access files using a built-in file handler from python.

The file will be read using the class' local 'file' variable:

```
f = open(self.file, "r", encoding="utf-8")
fContents = f.read()
```

The file contents will then be parsed by the decrypt function and loaded into memory using an SQLite method 'executescript()'.

## File Storage:

The program will store files using the same built-in file handler python provides.

First the program will create a variable containing the sql schema and data dump using sql's iterdump() method.

```
sqlStatement = ""
for line in conn.iterdump():
    sqlStatement += line
conn.close()
```

The system will then encrypt the data using the algorithm designed above. The encrypted data is finally stored as a custom extension text file: 'data.lzl' using python's file.write() where it can be parsed by the program again the next time it is executed.

## Query SQL table:

Executes an sql statement to search the class' table using each of the edited fields of the class' entries. It then returns the results in the treeview using the class' fillTable() function.

In this code **entryVars** is a local Ordered Dictionary per class that can be searched to find the user's input for each entry corresponding to a table column. This was decided because the program will modularly create the UI for each sql table according to the data that is stored in the table i.e. how many columns, column names, and column type.

```

FUNCTION searchQuery () :
    query = "SELECT * FROM " + table.name + " WHERE "
    numberOfEntries = len(entryVars)
    emptyVars = 0

    FOR x IN range(length of entryVars):
        key = list(entryVars) [x]
        value = entries[x].get()

        IF value == "" OR value == "None" THEN
            emptyVars += 1
        ELSE:
            query += key + " = \\" + value + "\ AND "

    # If there were no values entered
    IF emptyVars == numberOfEntries THEN
        OUTPUT "No values entered, displaying whole table"
        table.fillTable()
        RETURN
    # If there was a value entered
    ELSE:
        query = query[:-5]
        OUTPUT query
        table.fillTable(query)
        RETURN

    END IF

END FUNCTION

```

## Update SQL table:

Executes an sql statement to update the class' table using each of the edited fields of the class' entries. Similar to querying it then returns the results in the treeview using the class' fillTable() function.

```

FUNCTION updateQuery():
    sql = "UPDATE " + table.name + " SET "
    ## Select specific columns to add
    FOR i IN range (length of self.entryVars):
        value = entries[i].get()

        IF value NOT EMPTY STRING:
            sql = sql + columns[i] + " = '" + value + "', "
        END IF

    END FOR

    sql = sql[:-2] #remove comma and space at end of for loop
    sql += " WHERE id = '" + entryVars[columns[0]] + "'"
    c.execute(sql)
    c.close()
    table.fillTable() #update table

END FUNCTION

```

## Date Sort:

The system will require a method of sorting results of queries returning a data type of date. This algorithm will take a list of unsorted dates and sort them recursively according to year, then month, and finally date, then return the list with the sorted results.

```

FUNCTION mergeSortDates(dates):

    # Exit condition
    IF length of dates is equal to 1 THEN
        RETURN dates
    END IF

```

```

# Divide list into two halves
mid = floor(length of dates / 2)
left_dates = sublist of dates from 0 to mid-1
right_dates = sublist of dates from mid to end

# Recursively sort each half
left_dates = CALL mergeSortDates(left_dates)
right_dates = CALL mergeSortDates(right_dates)

# Merge the two halves together
sorted_dates = empty list
i = j = 0
WHILE i is less than length of left_dates AND j is less than
length of right_dates DO
    date = left_dates[i]
    today = right_dates[j]
    IF (date[2] < today[2] OR
        (date[2] == today[2] AND date[1] < today[1]) OR
        (date[2] == today[2] AND date[1] == today[1] AND date[0]
        < today[0])) THEN
        append date to sorted_dates
        i += 1
    ELSE
        append today to sorted_dates
        j += 1
    END IF
END WHILE

# Add any remaining elements in the left list
WHILE i is less than length of left_dates DO
    append left_dates[i] to sorted_dates
    i += 1
END WHILE

```

```

# Add any remaining elements in the right list
WHILE j is less than length of right_dates DO
    append right_dates[j] to sorted_dates
    j += 1
END WHILE

# Return the sorted list
RETURN sorted_dates
END FUNCTION

```

## **Integer Sort:**

The system will require a feature to sort results of table queries by integer for sorting ID's, prices, and other numbered results.

```

FUNCTION mergeSortInt(integers):

    # Exit condition
    IF length of integers is equal to 1 THEN
        RETURN integers
    END IF

    # Divide list into two halves
    mid = floor(length of integers / 2)
    left_integers = sublist of integers from 0 to mid-1
    right_integers = sublist of integers from mid to end

    # Recursively sort each half
    left_integers = CALL mergeSortInt(left_integers)
    right_integers = CALL mergeSortInt(right_integers)

    # Merge the two halves together
    sorted_integers = empty list
    i = j = 0
    WHILE i is less than length of left_integers AND j is less
    than length of right_integers DO

```

```
left_integer = left_integers[i]
right_integer = right_integers[j]
IF (int(left_integer[0]) < int(right_integer[0])) THEN
    append left_integer to sorted_integers
    i += 1
ELSE
    append right_integer to sorted_integers
    j += 1
END IF
END WHILE

# Add any remaining elements in the left list
WHILE i is less than length of left_integers DO
append left_integers[i] to sorted_integers
i += 1
END WHILE

# Add any remaining elements in the right list
WHILE j is less than length of right_integers DO
append right_integers[j] to sorted_integers
j += 1
END WHILE

# Return the sorted list
RETURN sorted_integers

END FUNCTION
```

## String Sort:

I will need a sorting algorithm to sort the results of queries by the ASCII codes of the characters in case the contents of the column does not match a previous sorting type.

```

FUNCTION mergeSortString(strings):

    # Exit condition
    IF length of strings is equal to 1 THEN
        RETURN strings
    END IF

    # Divide list into two halves
    mid = floor(length of strings / 2)
    left_strings = sublist of strings from 0 to mid-1
    right_strings = sublist of strings from mid to end

    # Recursively sort each half
    left_strings = CALL mergeSortString(left_strings)
    right_strings = CALL mergeSortString(right_strings)

    # Merge the two halves together
    sorted_strings = empty list
    i = j = 0
    WHILE i is less than length of left_strings AND j is less
    than length of right_strings DO
        left_string = left_strings[i]
        right_string = right_strings[j]

        IF (left_string[0] < right_string[0]) THEN
            append left_string to sorted_strings
            i += 1
        ELSE
            append right_string to sorted_strings
            j += 1
        END IF
    END WHILE

```

```

# Add any remaining elements in the left list
WHILE i is less than length of left_strings DO
    append left_strings[i] to sorted_strings
    i += 1
END WHILE

# Add any remaining elements in the right list
WHILE j is less than length of right_strings DO
    append right_strings[j] to sorted_strings
    j += 1
END WHILE

# Return the sorted list
RETURN sorted_strings

END FUNCTION

```

## Data Backup:

When saving the program will also attempt to make a backup. It will check the date of the last backup, and if it was more than 2 days ago it will make a new backup.

```

FUNCTION backupData():
    #Get the date and time of the last backup
    lastBackupDate = getDateOfLastBackup()

    #Get the current date and time
    currentDate = getCurrentDate()

    #Calculate the difference between the last backup date and the
    current date
    difference = currentDate - lastBackupDate

    #If the difference is greater than 2 days, create a new backup
    IF difference > 2 DAYS THEN
        backupDataToFile()
        OUTPUT "Data backed up successfully"
    ELSE

```

```
    OUTPUT "No backup needed"  
ENDIF  
END FUNCTION
```

# CALCULATIONS

## Estimate Monthly Student Revenue:

The manager may use a button to calculate the total income from student bookings that month. The calculation checks each booking that isn't cancelled and uses the booking frequency (how many weeks a month the student has a lesson), and lesson cost to calculate the overall income from students the company will receive (minus cancellations).

```
FUNCTION calculateStudentRevenue () :
    #Get selected rows
    curItems = self.table.tree.selection()

    #If selection is not empty
    IF len(curItems) != 0 THEN

        #define cost estimate text and monthly cost variables
        self.calculationEstimate = ""
        self.monthlyRevenue = 0

        #Clear the Text widget
        self.revenueFrame.text['state'] = "normal"
        self.revenueFrame.text.delete(1.0, END)

        #Loop through each selected row
        FOR item in curItems DO
            selectedItem = self.table.tree.item(item) ['values']
            #Calculate the monthly cost and the calculation
            estimate text
            monthlyCost = float(selectedItem[4]) *
            float(selectedItem[8])
            self.calculationEstimate += "User ID '" +
            str(selectedItem[0]) + "' has " + str(selectedItem[4]) + " monthly
            lessons, each costing £" + str(selectedItem[8]) + " resulting in a
            monthly revenue of: £" + str(monthlyCost) + "\n"
            self.monthlyRevenue += monthlyCost

            #Add the total monthly revenue to the text
            self.calculationEstimate += "\nTotal Monthly Estimate for
            Selected Lessons: £" + str(self.monthlyRevenue)

        #Set the Text() widget to display the information
```

```
        self.revenueFrame.text.insert(END,  
self.calculationEstimate)  
        self.revenueFrame.text['state'] = "disabled"  
    ELSE  
        print("Please Select a row")
```

# References

To help with the algorithms and design I looked at online sources. Following is a list of websites used to help me design the algorithms I will be using for the new system.

<https://cryptographyacademy.com/rsa/> - Website by the creators of the RSA algorithm, assisted me in designing the asymmetric encryption algorithm.

<https://stackoverflow.com/> - Source referenced to assist with recursive algorithms for sorting and encryption.

# Prototype

## FULL SPECIFICATION OBJECTIVES

### **1. User Access and Login System:**

1.1 - Restrict app to allow different levels of access for managers, employees, teachers, and students.

1.2 - Login system to authenticate users and grant appropriate permissions. Users will only have access to the features of the GUI their access levels allow.

### **2. Customer Records:**

2.1 - Managers can create, update, and delete validated customer records with their personal information.

2.2 - Display list of customers with personal information to managers in a searchable and sorted table.

2.3 - Display the purchase history of customers in a searchable and sorted table.

### **3. Employee Records:**

3.1 - Managers can create, update, and delete validated employee records with their personal information.

3.2 - Display list of employees with personal information to managers in a searchable and sorted table.

3.3 - Employees can edit their own records.

### **4. Teacher and Student Records:**

4.1 - Managers can create, update, and delete validated teacher and student profiles with their personal information.

4.2 - Display list of teachers and students with personal information to managers in a searchable and sorted table.

4.3 - Teachers and students can edit their own records

4.4 - Teachers and managers can view the progress and attendance of students in their respective lessons.

**5. Sales and Inventory Management:**

- 5.1 - Managers can create, update, and delete validated stock items for both the music store and the cafe with detailed information, such as product name, description, and price.
- 5.2 - Record inventory levels over a period of time.
- 5.3 - Track sales from user input.
- 5.4 - Automatically update inventory levels with sales.
- 5.5 - Allow managers to add items to inventory levels.

**6. Student Lessons:**

- 6.1 - Managers can create, update, and delete each student's lesson bookings.
- 6.2 - Managers and teachers can view searchable and sorted lists of student reports.
- 6.3 - Teachers can submit lesson reports after each lesson.
- 6.4 - Managers can create lesson plans.
- 6.5 - Managers and teachers can view a searchable and sorted list of lesson plans.

**7. Reporting and Data Analysis:**

- 7.1 - Calculate and display sales report given a date range.
- 7.2 - Calculate and display value of inventory.
- 7.3 - Display estimate of revenue from student lessons.
- 7.4 - Support exporting calculations to .txt or .csv format.

**8. Data Security and Integrity**

- 8.1 - Encrypt all stored data to prevent personal information being leaked.
- 8.2 - Make routine backups of all stored data.
- 8.3 - Validate all user input.

# Prototype Planning

## PRE-PROTOTYPE INTRODUCTION

A prototype containing samples of each feature the final software requires will be produced to present to the manager of Lizzy Lee's. The software will be created with a priority on features and not a focus on the (visual) design or efficiency of code as this can be optimised after the features are all implemented.

The result of the prototype will be presented to the manager, whose feedback will be considered to alter the design before the final product is developed.

The prototype should provide at the least one instance of every *similar* feature required by the stakeholders. The purpose of the prototype is to prove that the final product will be functional, useful, and feasible.

For instance, in the record section displayed in the design document there will be a tab for the employees, teachers, students, and customers. Each tab will have the same design and be programmed identically, so the prototype will not need to demonstrate each tab.

## OBJECTIVES

Here I will specify what procedures should be included in the prototype and what procedures can be omitted.

### **Login and Level Access:**

The Prototype should demonstrate a functional login system using details from the SQL document that determines the users' permitted access levels and restrict access based on this. Not each access level needs to be demonstrated, but at least two should be.

### **Records:**

At least 2 personal record sections should be fully functional to demonstrate how the features will function and the method of navigation between record interfaces.

**Student Lessons:**

The section with student lessons should be functional. Lesson reports and plans do not need to be demonstrated as they function similarly to the already demonstrated personal records and student lesson tabs.

**Sales and Stock:**

The prototype should include an instance of the sales and stock system working in conjunction. They should be displayed in their own section as demonstrated in the design, and should provide a demonstration of the sales system affecting stock levels.

**Data Export and Import:**

A simple window design should demonstrate the feature of exporting and importing data. The windows will not need to have a nice design, but should focus on function.

**Security:**

The prototype will need to include examples of the security features the system prioritises. This includes **validation, encryption, backups, and password protection**.

**Data Analysis:**

The prototype should also include a working instance of the data analysis features that will increase user productivity such as record sorting and filtering.

# Prototype Report

For each objective in the plan I will report how the prototype has achieved or fell short of achieving them.

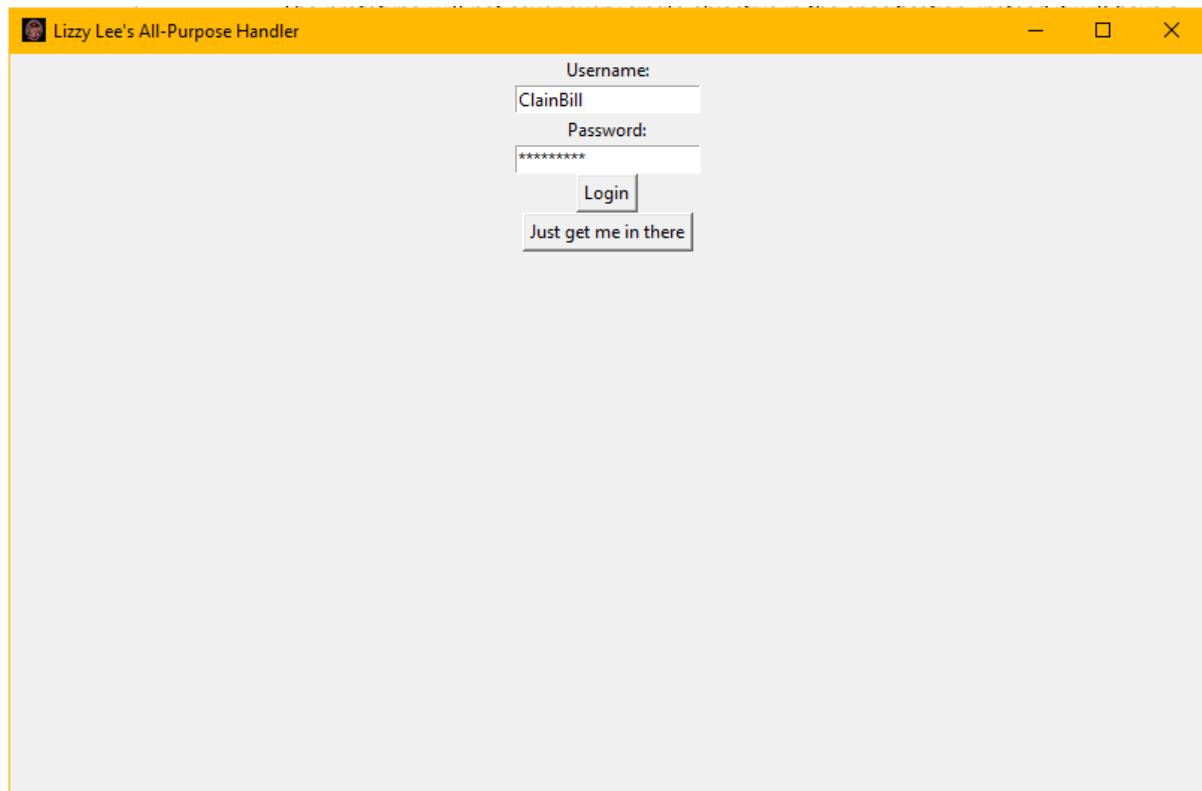
## 1. User Access and Login System:

1.1 - *Restrict app to allow different levels of access for managers, employees, teachers, and students.*

The prototype does not restrict layers of access according to the user's access rights. However the application does record what access rights users have according to the user's table name or job role (if they are an employee). With this feature available it will be made simpler to implement access levels in the complete application.

I will do this by giving each class a variable that determines what access levels will display that class. When the user logs in, classes will check for the user's access level and hide themselves if the access level attribute does not match the user's access level.

1.2 - *Login system to authenticate users and grant appropriate permissions. Users will only have access to the features of the GUI their access level allows.*



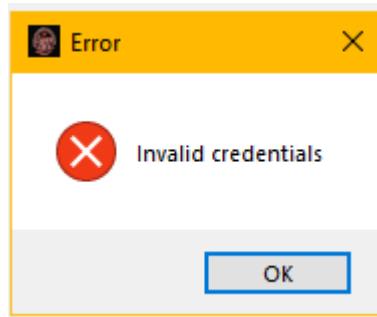
The system has a fully functional login screen that accepts correct credentials and rejects incorrect attempts. It iterates over each table in memory with a username and password column, and compares the normalised username field and the password field.

The system checks the user's role by retrieving the table name the credentials match. If the table is an EMPLOYEE table it then checks for the user's jobRole value within the table.

With this role the system can restrict access to system features or allow other features.

The displayed login screen also demonstrates a second button designed exclusively for the prototyping stage of development. This button skips the login and assumes the role of 'Manager' to speed up the debugging and testing stage. The button will be removed from the final product.

---



Incorrect login attempts will trigger a system warning message, the user can then try again with a new username and password.

As stated above, when login attempts are successful the system will give every user access to the same GUI screen after logging in. The only difference will be the title of the window which will display the role the user has.

## 2. Customer Records:

The “Customer Records” tab was not included in the prototype, but it functions identically as the other person records that have been included.

2.3 - *Display the purchase history of customers in a searchable and sorted table.*

This will be identical to the sales section but will only display sales related to this specific customer in the same way that customers can only see their own record.

## 3. Employee Records:

3.1 - *Managers can create, update, and delete validated employee records with their personal information.*

When the user logs in they are sent to the employee record tab:

The screenshot shows a Windows application window titled "Lizzy Lee's - Manager Dashboard". The menu bar includes "File", "Edit", and "Account". On the left, there are three buttons: "Records" (highlighted), "Booking", and "Stock". The main area contains a table titled "Employee" with columns: ID, username, forename, surname, sex, title, birthdate, hire\_date, job\_description, and town. The table has 16 rows of data. Below the table is a "Record" form with fields for ID, username, password, forename, surname, sex, title, birthdate, hire\_date, job\_description, town, county, postcode, phone\_num, and email. At the bottom, there is a "Commands" bar with buttons: Clear Fields, Search, Update, Add Record, and Delete Record.

ID	username	forename	surname	sex	title	birthdate	hire_date	job_description	town
1	mdavies	Megan	Davies	F	Mrs	19/10/1990	02/01/2011	Manager	Benllech
2	nhedd	Nia	Hedd	M	Ms	10/08/1987	04/04/2011	Manager	Llanbadrig
3	mlloyd	Megan	Lloyd	M	Miss	16/04/1985	23/06/2010	Manager	Llanbadrig
4	aidris	Aled	Idris	M	Miss	06/05/1986	11/01/2011	Manager	Pentraeth
5	tthomas	Tegan	Thomas	M	Mrs	09/02/1998	14/05/2015	Crew	Menai Bridge
6	ifychan	Iwan	Fychan	F	Dr	23/01/1996	27/07/2013	Crew	Moelfre
7	mbevan	Megan	Bevan	F	Dr	25/01/1985	15/10/2011	Crew	Menai Bridge
8	udavies	Urien	Davies	M	Ms	15/12/1989	10/06/2012	Crew	Newborough
9	hwalters	Huw	Walters	F	Mr	20/05/1975	04/10/2013	Crew	Beaumaris
10	bhedd	Bryn	Hedd	F	Mr	17/08/1975	26/06/2018	Crew	Holyhead
11	nceri	Nia	Ceri	F	Miss	24/09/1983	15/12/2012	Crew	Pentraeth
12	nceri	Nia	Ceri	F	Ms	28/10/1993	06/06/2016	Crew	Bodedern
13	uidris	Urien	Idris	F	Dr	19/05/1976	26/07/2019	Manager	Cemaes
14	gnia	Gareth	Nia	M	Mrs	12/10/1986	25/04/2021	Manager	Llanbadrig
15	howen	Huw	Owen	M	Mrs	13/07/1966	02/06/2010	Manager	Rhosneigr
16	iidris	Iwan	Idris	F	Mrs	04/08/1987	07/12/2012	Manager	Holyhead

The centre frame displays a list of all the employees' records. Buttons at the bottom provide the ability to update, create, and delete records.



As can be seen above the user that logged in has the access level 'crew', but is able to create, update, and delete records which only the manager should be able to do.

This is something that will be addressed in the final product.

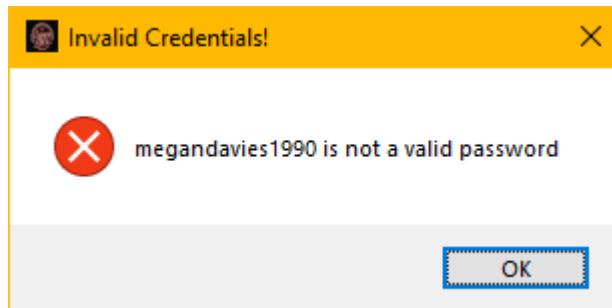
When a record is selected the entry fields are populated with information from the record:

ID	username	forename	surname	sex	title	birthdate	hire_date	job_description	town
1	mdavies	Megan	Davies	F	Mrs	19/10/1990	02/01/2011	Manager	Benllech
2	nhedd	Nia	Hedd	M	Ms	10/08/1987	04/04/2011	Manager	Llanbadrig
3	milloyd	Megan	Lloyd	M	Miss	16/04/1985	23/06/2010	Manager	Llanbadrig
4	aidris	Aled	Idris	M	Miss	06/05/1986	11/01/2011	Manager	Pentraeth
5	tthomas	Tegan	Thomas	M	Mrs	09/02/1998	14/05/2015	Crew	Menai Bridge
6	ifychan	Iwan	Fychan	F	Dr	23/01/1996	27/07/2013	Crew	Moelfre
7	mbevan	Megan	Bevan	F	Dr	25/01/1985	15/10/2011	Crew	Menai Bridge
8	udavies	Urien	Davies	M	Ms	15/12/1989	10/06/2012	Crew	Newborough
9	hwalters	Huw	Walters	F	Mr	20/05/1975	04/10/2013	Crew	Beaumaris
10	bhedd	Bryn	Hedd	F	Mr	17/08/1975	26/06/2018	Crew	Holyhead
11	nceri	Nia	Ceri	F	Miss	24/09/1983	15/12/2012	Crew	Pentraeth
12	nceri	Nia	Ceri	F	Ms	28/10/1993	06/06/2016	Crew	Bodedern
13	uidris	Urien	Idris	F	Dr	19/05/1976	26/07/2019	Manager	Cemaes
14	gnia	Gareth	Nia	M	Mrs	12/10/1986	25/04/2021	Manager	Llanbadrig
15	howen	Huw	Owen	M	Mrs	13/07/1966	02/06/2010	Manager	Rhosneigr
16	iidris	Iwan	Idris	F	Mrs	04/08/1987	07/12/2012	Manager	Holyhead

These entries can be edited and then using the update button the entries will be validated then update their corresponding SQL record.

Commands				
<a href="#">Clear Fields</a>	<a href="#">Search</a>	<a href="#">Update</a>	<a href="#">Add Record</a>	<a href="#">Delete Record</a>

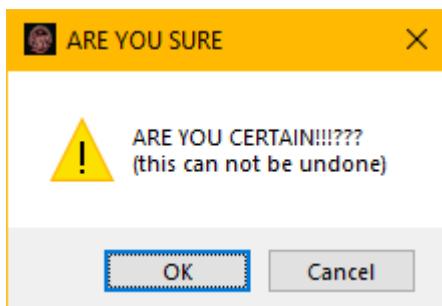
If an entry is invalid (section 8.3) the system will throw an error:



Otherwise a confirmation will appear:



When deleting a record the system warns the user:



Where they will be presented with a second chance to cancel the operation if they are unsure, or persist and delete the record from this save. Backups can be used to view old data and restore lost data if an accident still occurs (section 8.2).

Finally, users can add a record to the database by clicking the "Add Record" button

This acts identically to updating records, but will create a new record instead of updating an existing one.

3.2 - Display list of employees with personal information to managers in a searchable and sorted table.

All records are placed in the treeview list. If the list of records exceeds the view vertically or horizontally the user can scroll using a mousewheel or the scrollbars on either edge of the treeview frame.

The entry list in the “Record” section of the tab can also be used to filter the treeview results. When the user clicks on “Search” in the “Commands” section the system will check each entry with values filled in and will compare these fields with data in the database.

A list of records matching the filter will be returned in the treeview frame. For instance, if only the “Title” field is filled with “Dr” the “Search” button will fill the treeview with:

The screenshot shows a Windows application window titled "Lizzy Lee's - Manager Dashboard". The menu bar includes "File", "Edit", and "Account". The main area features a treeview with three nodes: "Records", "Booking", and "Stock". The "Records" node is expanded, displaying a table of employee data with columns: ID, username, forename, surname, sex, title, birthdate, hire\_date, job\_description, and town. The table contains 5 rows of data. Below the treeview is a "Record" form with fields for ID, username, password, forename, surname, sex, title, birthdate, hire\_date, job\_description, town, county, postcode, phone\_num, and email. At the bottom of the window is a "Commands" toolbar with buttons for Clear Fields, Search, Update, Add Record, and Delete Record.

ID	username	forename	surname	sex	title	birthdate	hire_date	job_description	town
6	ifychan	Iwan	Fychan	F	Dr	23/01/1996	27/07/2013	Crew	Moelfre
7	mbevan	Megan	Bevan	F	Dr	25/01/1985	15/10/2011	Crew	Menai Bridge
13	uidris	Urien	Idris	F	Dr	19/05/1976	26/07/2019	Manager	Cemaes
17	sidris	Sian	Idris	M	Dr	21/12/1986	12/05/2019	Manager	Moelfre
20	kidris	Katie	Idris	M	Dr	09/10/1981	13/05/2022	Crew	Cemaes

Furthermore, multiple filters can be used simultaneously:

ID	username	forename	surname	sex	title	birthdate	hire_date	job_description	town
6	ifychan	Iwan	Fychan	F	Dr	23/01/1996	27/07/2013	Crew	Moelfre
7	mbevan	Megan	Bevan	F	Dr	25/01/1985	15/10/2011	Crew	Menai Bridge
20	kidris	Katie	Idris	M	Dr	09/10/1981	13/05/2022	Crew	Cemaes

In addition to filtering results users can sort results by any of the columns. This is intuitively achievable by clicking on the column header of the column the user wishes to sort by.

Strings are sorted by alphabetical order, numbers are sorted by magnitude, and dates are sorted by date.

#### Unsorted (default is ID):

ID	username	forename	surname	sex	title	birthdate
6	ifychan	Iwan	Fychan	F	Dr	23/01/1996
7	mbevan	Megan	Bevan	F	Dr	25/01/1985
20	kidris	Katie	Idris	M	Dr	09/10/1981

#### Sorted by Forename:

ID	username	forename	surname	sex	title	birthdate
7	mbevan	Megan	Bevan	F	Dr	25/01/1985
6	ifychan	Iwan	Fychan	F	Dr	23/01/1996
20	kidris	Katie	Idris	M	Dr	09/10/1981

#### Sorted by Date:

ID	username	forename	surname	sex	title	birthdate
20	kidris	Katie	Idris	M	Dr	09/10/1981
7	mbevan	Megan	Bevan	F	Dr	25/01/1985
6	ifychan	Iwan	Fychan	F	Dr	23/01/1996

The headers can then be clicked again to reverse the sorting order:

Reversed Date:

ID	username	forename	surname	sex	title	birthdate
6	ifychan	Iwan	Fychan	F	Dr	23/01/1996
7	mbevan	Megan	Bevan	F	Dr	25/01/1985
20	kidris	Katie	Idris	M	Dr	09/10/1981

3.3 - Employees can edit their own records.

The prototype does not distinguish between user logins. All users who successfully login can edit any record like a manager (so this objective is technically fulfilled 😊 )

The final product will use access levels to only display the employees own record using the filter function already demonstrated. Employees will be locked to this filter so they can only view and edit their own record.

#### 4. Teacher and Student Records:

4.1 - Managers can create, update, and delete validated teacher and student profiles with their personal information.

This objective works identically to the employee records in 3.1. The record fields are dynamically set to associate with each column in the Teacher and Student tabs respectively:

##### Teacher Tab:

Employee	Teacher	Student							
1	town	Tegan	Owen	M	Ms	12/03/1991	10/12/2021	Menai Bridge	Conwy
2	bprice	Bryn	Price	F	Miss	08/08/1987	21/03/2018	Moelfre	Anglesey
3	troberts	Tegan	Roberts	M	Dr	18/11/1983	25/06/2013	Llandegfan	Gwynedd
4	jnia	Jade	Nia	M	Ms	20/07/1961	06/04/2016	Trearddur Bay	Gwynedd
5	llloyd	Llion	Lloyd	F	Mr	17/10/1998	19/12/2016	Llanddona	Gwynedd
6	ewilliams	Elin	Williams	M	Mr	27/04/1996	17/09/2018	Trearddur Bay	Gwynedd
7	fowen	Ffion	Owen	F	Miss	22/10/1995	13/11/2014	Newborough	Gwynedd
8	onia	Owain	Nia	F	Mr	08/06/1970	20/06/2012	Newborough	Gwynedd
9	mrees	Megan	Rees	F	Mr	13/06/1986	22/08/2014	Llanerchymedd	Anglesey
10	uwilliams	Urien	Williams	M	Mr	03/05/1970	20/06/2020	Benllech	Conwy
11	sfychan	Sian	Fychan	F	Ms	14/03/1997	19/04/2017	Y Felinheli	Anglesey
12	ghedd	Gareth	Hedd	F	Mr	22/02/1984	04/04/2020	Beaumaris	Anglesey
13	jroberts	Jade	Roberts	F	Mr	16/09/1980	19/07/2014	Moelfre	Anglesey
14	uyorath	Urien	Yorath	F	Mrs	18/02/1997	01/08/2018	Y Felinheli	Conwy
15	rjones	Rhian	Jones	F	Mr	05/08/1989	05/08/2015	Cemaes	Conwy
16	bprice	Bryn	Price	F	Miss	20/11/1990	21/09/2016	Rhosneigr	Gwynedd

Record

ID:	<input type="text"/>	username:	<input type="text"/>	password:	<input type="text"/>
forename:	<input type="text"/>	surname:	<input type="text"/>	sex:	<input type="text"/>
title:	<input type="text"/>	birthdate:	<input type="text"/>	hire_date:	<input type="text"/>
town:	<input type="text"/>	county:	<input type="text"/>	postcode:	<input type="text"/>
phone_num:	<input type="text"/>	email:	<input type="text"/>		

Commands

[Clear Fields](#) [Search](#) [Update](#) [Add Record](#) [Delete Record](#)

##### Student Tab:

Employee	Teacher	Student							
2	Sioned786	Nia	Davies	F	Mr	1/1/1999	5/10/2010	Llangefni	Anglesey
3	Cadell129	Branwen	Morgan	M	Ms	14/3/1986	14/12/2015	Benllech	Gwynedd
4	Mabon420	Nia	Edwards	F	Mrs	14/8/1981	16/3/2015	Menai Bridge	Conwy
5	Haf792	Llio	Khan	M	Mr	26/6/1988	26/12/2014	Llangefni	Anglesey
6	Haf259	Ffion	Williams	F	Mrs	4/4/1997	21/11/2013	Amlwch	Conwy
7	Owain856	Haf	Bevan	M	Mr	22/8/1992	4/7/2016	Benllech	Gwynedd
8	Ynry627	Haf	Ceredig	F	Mrs	16/6/2000	28/4/2011	Benllech	Conwy
9	Eira336	Rhian	Howells	F	Mr	17/1/2002	3/2/2016	Pentraeth	Anglesey
10	Llio584	Nia	Ap Dafydd	M	Miss	4/1/1995	15/11/2012	Benllech	Gwynedd
11	Cadell837	Zara	Price	F	Mrs	23/3/1993	12/1/2015	Benllech	Conwy
12	Dyfan850	Ffion	Fychan	F	Mr	9/10/1996	13/10/2021	Rhosneigr	Anglesey
13	Haf379	Wyn	Ceredig	M	Mr	5/7/1983	20/7/2011	Benllech	Anglesey
14	Iestyn641	Rhian	Bevan	F	Miss	4/11/1996	3/11/2021	Rhosneigr	Gwynedd
15	Branwen223	Sioned	Lloyd	M	Mr	9/6/2001	20/3/2011	Benllech	Conwy
16	Ynry776	Ffion	Williams	F	Mrs	20/9/1984	27/3/2022	Menai Bridge	Anglesey
17	Nia763	Gethin	Fychan	M	Mrs	16/4/1999	11/1/2017	Gaerwen	Anglesey

Record

ID:	<input type="text"/>	username:	<input type="text"/>	password:	<input type="text"/>
forename:	<input type="text"/>	surname:	<input type="text"/>	sex:	<input type="text"/>
title:	<input type="text"/>	birthdate:	<input type="text"/>	hire_date:	<input type="text"/>
town:	<input type="text"/>	county:	<input type="text"/>	postcode:	<input type="text"/>
phone_num:	<input type="text"/>	guardian_phone_num:	<input type="text"/>	email:	<input type="text"/>

Commands

[Clear Fields](#) [Search](#) [Update](#) [Add Record](#) [Delete Record](#)

4.2 - Display list of teachers and students with personal information to managers in a searchable and sorted table.

As you can see from the images above (4.1) the teacher and student records display a searchable list of data containing every record. The scrollbars are activated if the number of records exceeds the frame size.

The screenshot shows a Windows application window titled "Lizzy Lee's - Manager Dashboard". The menu bar includes "File", "Edit", and "Account". On the left, there are four buttons: "Records", "Lessons", "Stock", and "Sales". The main area contains a table with 16 rows of data, each representing a record with columns for ID, username, forename, surname, sex, title, birthdate, hire\_date, town, and county. Below the table is a "Record" form with fields for ID, username, password, forename, surname, sex, title, birthdate, hire\_date, town, county, postcode, phone\_num, and email. At the bottom, there is a "Commands" section with buttons for Clear Fields, Search, Update, Add Record, and Delete Record.

ID	username	forename	surname	sex	title	birthdate	hire_date	town	county
1	towen	Tegan	Owen	M	Ms	12/03/1991	10/12/2021	Menai Bridge	Conwy
2	bprice	Bryn	Price	F	Miss	08/08/1987	21/03/2018	Moelfre	Anglesey
3	troberts	Tegan	Roberts	M	Dr	18/11/1983	25/06/2013	Llandegfan	Gwynedd
4	jnia	Jade	Nia	M	Ms	20/07/1961	06/04/2016	Trearddur Bay	Gwynedd
5	llloyd	Llion	Lloyd	F	Mr	17/10/1998	19/12/2016	Llanddona	Gwynedd
6	ewilliams	Elin	Williams	M	Mr	27/04/1996	17/09/2018	Trearddur Bay	Gwynedd
7	fowen	Ffion	Owen	F	Miss	22/10/1995	13/11/2014	Newborough	Gwynedd
8	onia	Owain	Nia	F	Mr	08/06/1970	20/06/2012	Newborough	Gwynedd
9	mrees	Megan	Rees	F	Mr	13/06/1986	22/08/2014	Llanerchymedd	Anglesey
10	uwilliams	Urien	Williams	M	Mr	03/05/1970	20/06/2020	Benllech	Conwy
11	sfychan	Sian	Fychan	F	Ms	14/03/1997	19/04/2017	Y Felinheli	Anglesey
12	ghedd	Gareth	Hedd	F	Mr	22/02/1984	04/04/2020	Beaumaris	Anglesey
13	jroberts	Jade	Roberts	F	Mr	16/09/1980	19/07/2014	Moelfre	Anglesey
14	uyorath	Urien	Yorath	F	Mrs	18/02/1997	01/08/2018	Y Felinheli	Conwy
15	rjones	Rhian	Jones	F	Mr	05/08/1989	05/08/2015	Cemaes	Conwy
16	bprice	Bryn	Price	F	Miss	20/11/1990	21/09/2016	Rhosneigr	Gwynedd

**Record**

ID:	<input type="text"/>	username:	<input type="text"/>	password:	<input type="text"/>
forename:	<input type="text"/>	surname:	<input type="text"/>	sex:	<input type="text"/>
title:	<input type="text"/>	birthdate:	<input type="text"/>	hire_date:	<input type="text"/>
town:	<input type="text"/>	county:	<input type="text"/>	postcode:	<input type="text"/>
phone_num:	<input type="text"/>	email:	<input type="text"/>		

**Commands**

The data is searchable and sorted using the same rules as demonstrated in **3.2**.

For example, out of 200 records the system instantly filtered out all the teachers who live in Llangefni and use the title “Mr.”. The results are sorted by “Forename”

Employee	Teacher	Student	ID	username	forename	surname	sex	title	birthdate	hire_date	town	county
			37	Aneirin750	Aneirin	Lloyd	F	Mr	3/12/1990	14/2/2013	Llangefni	Gwynedd
			133	Ffion367	Dyfan	Owen	M	Mr	12/08/1977	18/11/1960	Llangefni	Anglesey
			56	Branwen217	Ffion	Neville	M	Mr	7/4/1999	9/4/2019	Llangefni	Conwy
			5	Haf792	Llio	Khan	M	Mr	26/6/1988	26/12/2014	Llangefni	Anglesey
			52	Iestyn985	Mabon	Iorwerth	F	Mr	27/8/1982	14/3/2011	Llangefni	Gwynedd
			2	Sioned786	Nia	Davies	F	Mr	1/1/1999	5/10/2010	Llangefni	Anglesey
			31	Ynyr391	Sioned	Stephens	F	Mr	21/11/1984	4/5/2020	Llangefni	Anglesey
			177	Mabon771	Tegan	Ap Dafydd	F	Mr	15/02/1962	23/11/1974	Llangefni	Anglesey

Record

ID:	username:	password:	
forename:	surname:	sex:	
title:	birthdate:	hire_date:	
town:	county:	postcode:	
phone_num:	guardian_phone_num:	email:	

#### 4.3 - Teachers and students can edit their own records

This objective has not been implemented in the prototype for the same reason as is explained in section **3.3**. Additionally, it is worth noting that Students will only have access to the student tab and no other tab or section as it is irrelevant to them.

#### 4.4 - Teachers and managers can view the progress and attendance of students in their respective lessons.

The system displays a list of student reports that can be searched similarly to the implementation of human records. The lesson reports contain all the information needed to examine students

This will be explored further in section **6**.

## 5. Sales and Inventory Management:

5.1 - Managers can create, update, and delete validated stock items for both the music store and the cafe with detailed information, such as product name, description, and price.

In the “Stock” section the prototype contains tabs for music items and cafe items. These tabs provide the functionality for the user to create, edit, or delete stock items. Again, this works very similarly to the other procedures demonstrated where the user can select a record and edit its fields or create a new record using the entry widgets displayed below the treeview.:

### Music Items:

ID	brand	type	model	price	serial_num
1	Epiphone	Electric	Stratocaster	1056.01	TIMWXTGUSS
2	Fender	Bass	Flying V	614.02	S69DAIF1N4
3	Schecter	Acoustic	JEM	1104.66	OUGGEITVZV
4	Jackson	Acoustic	Telecaster	239.31	P6DQGHZ338
5	PRS	Electric	Stratocaster	986.61	2MVEFYEOGX
6	ESP	Classical	Flying V	77.57	LXUSSTCPB5
7	Ibanez	Electric	Flying V	1268.98	L8H97OGZ7P
8	Ibanez	Classical	Flying V	1717.84	ZZQHVDIMS7
9	ESP	Bass	Telecaster	1183	3N38MI9GQ3
10	Fender	Acoustic	Stratocaster	1202.72	2IUECSM223
11	Fender	Electric	Telecaster	1035.28	Z8YMSUAN38
12	Epiphone	Acoustic	JEM	230.89	8QCD601GD7
13	PRS	Electric	Telecaster	1612.01	DUIZ35WFBCB
14	ESP	Classical	Warlock	1345.06	SEQ2COXPPP
15	PRS	Acoustic	Flying V	762.43	GCL30LPKMW
16	PRS	Electric	JEM	1397.89	13NQP15SE8

The fields contain all the necessary fields to identify the product and also supports an optional “serial number” if the user needs to identify a specific item.

Cafe Items:

ID	type	description	price	expiry_date
1	Cocktail	Burger with fries	7.54	04/06/2023
2	Coffee	Brownie	9.64	01/04/2024
3	Cocktail	Tea with milk	9.94	28/03/2024
4	Wine	Stout	7.85	02/06/2023
5	Salad	Cola	4.53	27/07/2023
6	Soft drink	Tea with milk	9.5	10/07/2023
7	Beer	Merlot	6.28	06/04/2024
8	Wrap	Caesar salad	7.03	16/12/2023
9	Burger	Mocha	7.59	26/01/2024
10	Salad	Club sandwich	10.01	28/10/2023
11	Salad	Chicken wrap	14.31	04/10/2023
12	Cocktail	Cappuccino	9.11	22/02/2024
13	Dessert	Chardonnay	9.71	26/11/2023
14	Tea	Lemonade	13.82	07/11/2023
15	Pizza	Cappuccino	14.35	04/10/2023
16	Sandwich	Merlot	7.44	25/12/2023

The cafe items have an expiry date in place of a serial number as the system can be sorted to inform the user of when a product is set to expire.

5.2 - Record inventory levels over a period of time.

The system tracks inventory by taking a count of each item on that date:

Music Item	Cafe Item	Music Inventory	Cafe Inventory
ID	MusicItemID	count	date
96	50	5	01/11/2000
90	31	9	16/01/2000
98	19	8	09/08/1999
58	25	7	28/02/1999
89	44	10	14/02/1999
5	19	1	19/11/1998
2	27	10	22/08/1998
91	24	8	01/11/1997
61	1	9	28/08/1997
6	8	3	18/08/1997
42	45	9	06/07/1997
25	38	10	12/12/1996
83	42	10	05/10/1996
87	36	5	10/09/1996
36	35	4	05/05/1996
12	49	9	05/12/1995

Music Item	Cafe Item	Music Inventory	Cafe Inventory
ID	CafeItemID	count	date
1	40	6	19/09/1994
2	6	2	22/02/1964
3	33	9	16/02/1961
4	36	8	14/10/1960
5	47	4	21/08/1974
6	18	4	11/05/1993
7	38	2	24/06/1982
8	13	6	16/01/1977
9	6	5	24/01/1963
10	39	4	15/02/1978
11	19	2	05/07/1973
12	46	4	10/01/1964
13	23	5	23/08/1979
14	8	4	06/10/1986
15	40	9	16/09/1977
16	21	8	27/04/1995

The prototype can provide an estimated value of the inventory. The inventory estimate is calculated from the records selected:

ID	MusicItemID	count	date
1	7	3	21/07/1971
2	27	10	22/08/1998
3	45	8	06/07/1972
4	33	3	22/10/1968
5	19	1	19/11/1998
6	8	3	18/08/1997
7	28	6	09/10/1969
8	26	4	04/03/1980
9	31	9	17/10/1966
10	7	8	08/07/1974
11	11	9	15/11/1978
12	49	9	05/12/1995
13	25	6	05/11/1978
14	32	3	28/08/1961
15	22	8	21/01/1984
16	15	4	21/04/1970

Inventory Estimate  
Estimated inventory value: £10628.1

Record  
ID: 1      MusicItemID: 7      count: 3  
date: 21/07/1971

Commands  
Clear Fields | Search | Update | Add Record | Delete Record

When records are selected the system automatically calculates the inventory value by multiplying the cost of the associated product with the number of that item held. The results are displayed in the frame on the right.

### 5.3 - Track sales from user input.

The sales system works similarly to the inventory management system. Sales are tracked by inputting an item id, amount, and date. If the customer has an ID their ID can optionally be recorded as well.

ID	MusicItemID	CustomerID	count	date
1	44	46	8	21/03/1993
2	40	48	9	03/01/1997
3	30	1	7	05/12/1963
4	40	44	1	02/06/1989
5	44	35	4	06/04/1971
6	1	4	2	05/06/1977
7	49	23	8	04/01/1992
8	22	35	6	17/03/1996
9	23	16	1	18/10/1971
10	5	37	10	21/03/1979
11	14	40	7	18/12/1987
12	43	50	4	26/02/1977
13	41	22	2	26/04/1963
14	35	43	10	11/09/1985
15	41	16	1	01/06/1971
16	29	18	4	06/01/1984

Sales Estimate  
Total Revenue: £22435.21

Record  
ID: 1      MusicItemID: 44      CustomerID: 46  
count: 8      date: 21/03/1993

Commands  
Clear Fields | Search | Update | Add Record | Delete Record

*5.4 - Automatically update inventory levels with sales.*

The prototype does not automatically update inventory levels according to sales. This will need to be implemented in the final product by displaying the difference between levels of inventory stored and sales recorded.

*5.5 - Allow managers to add items to inventory levels.*

The user can add items to the table using the entries below and the “Add Record” button:

The screenshot shows a software interface with a header bar containing four input fields: ID (101), CafeltemID (12), count (2), and date (05/05/2023). Below this is a 'Record' section with three input fields: ID (101), CafeltemID (12), and count (2). Underneath is a date field with the value 05/05/2023. At the bottom is a 'Commands' section with five buttons: Clear Fields, Search, Update, Add Record (which is highlighted in blue), and Delete Record.

Here 2 of the cafe item “Medium Coca-Cola” was recorded in the cafe inventory on the 5th of May 2023

We can tell that it was Coca-Cola from looking at the cafe items table at ID 12:

Music Item	Cafe Item	Music Inventory	Cafe Inventory	
	ID	type	description	price
	2	Coffee	Brownie	9.64
	3	Cocktail	Tea with milk	9.94
	4	Wine	Stout	7.85
	5	Salad	Cola	4.53
	6	Soft drink	Tea with milk	9.5
	7	Beer	Merlot	6.28
	8	Wrap	Caesar salad	7.03
	9	Burger	Mocha	7.59
	10	Salad	Club sandwich	10.01
	11	Salad	Chicken wrap	14.31
	12	Soft drink	Medium Coca Cola	1.25
	13	Dessert	Chardonnay	9.71
	14	Tea	Lemonade	13.82
	15	Pizza	Cappuccino	14.35
	16	Sandwich	Merlot	7.44
	17	Burger	Latte	10.56

## 6. Student Lessons:

6.1 - Managers can create, update, and delete each student's lesson bookings.

The prototype supports the creation and maintenance of lesson bookings using the same system as demonstrated in 3.3.

ID	StudentID	TeacherID	location	lesson_frequency	lesson_day
1	93933250	97904097	Room 10	Fortnightly	Wednesday
2	23078257	20588786	Room 5	Monthly	Friday
3	35794237	74234495	Room 9	Weekly	Tuesday
4	37601708	23868595	Room 5	Fortnightly	Friday
5	42880367	30548433	Room 6	Fortnightly	Monday
6	58206941	51551392	Room 10	Fortnightly	Monday
7	36918344	23862463	Room 7	Fortnightly	Thursday
8	55479729	36566107	Room 10	Monthly	Wednesday
9	34505342	81151522	Room 9	Fortnightly	Monday
10	89365844	49711713	Room 9	Fortnightly	Monday
11	57739386	49852907	Room 4	Weekly	Wednesday
12	11488486	35756230	Room 9	Monthly	Tuesday
13	83471402	55583555	Room 8	Weekly	Thursday
14	76380853	98492717	Room 9	Monthly	Monday
15	65557280	20327883	Room 10	Monthly	Wednesday
16	39544820	97437566	Room 2	Weekly	Monday

The tab additionally shows a text field containing an estimate of the revenue generated by the selected student bookings in a month. It calculates the cost from looking at the lesson frequency, lesson cost, and whether or not the lesson has been cancelled

```
monthlyCost = lessonCost * lessonFrequency * notCancelled
```

Multiple lessons can be selected to gain an estimate for a number of students or teachers:

TeacherID	location	lesson_frequency	lesson_day	lesson_time	lesson_length
55583555	Room 8	Weekly	Thursday	15:47	60
96489000	Room 1	Weekly	Thursday	15:51	45
61427500	Room 6	Fortnightly	Thursday	15:22	45
53056983	Room 6	Fortnightly	Thursday	09:53	60
96544155	Room 2	Fortnightly	Thursday	08:51	30
38403113	Room 6	Weekly	Thursday	16:44	30
99451196	Room 1	Monthly	Thursday	10:26	45
81798957	Room 6	Fortnightly	Thursday	14:35	30
59964335	Room 8	Monthly	Thursday	08:16	30
36560093	Room 1	Monthly	Thursday	09:15	60
55106616	Room 4	Monthly	Thursday	11:17	60
58237581	Room 10	Monthly	Thursday	09:04	30
16582659	Room 9	Weekly	Thursday	15:00	60

User ID '13' has Weekly monthly lessons, each costing £10 result ^  
User ID '23' has Weekly monthly lessons, each costing £15 result  
User ID '29' has Fortnightly monthly lessons, each costing £15 result  
User ID '90' has Weekly monthly lessons, each costing £20 result  
Total Monthly Estimate for Selected Lessons: £210.0

This image demonstrates looking at the revenue generated from student lessons on Thursdays at 3pm

## 6.2 - Managers and teachers can view searchable and sorted lists of student reports.

Student reports are listed in a treeview and are manageable similarly to the other records, however the report notes of the selected record are also displayed in a separate text frame:

ID	LessonPlansID	StudentID	TeacherID	attended	student_behav
1	50	4	32	0	1
2	35	32	48	0	2
3	45	22	24	1	0
4	37	3	40	0	0
5	49	26	33	1	0
6	33	25	4	1	0
7	34	35	46	1	2
8	42	45	31	1	1
9	29	3	50	0	0
10	49	5	27	0	4
11	5	21	21	0	0
12	15	25	17	1	3
13	41	10	27	0	2
14	10	47	35	0	0
15	11	12	32	1	1
16	19	46	3	1	3

Report Notes  
Huw was able to comprehend the lesson material very intuitively ^  
Huw wants to learn about chord progressions soon

Note Commands  
Make changes to note:

This is to account for readability. Here the user can read the notes in a manageable format and can edit them from the frame if necessary.

The student reports can be searched similarly to the employee records in **3.2** however the access levels have not yet been implemented to restrict teachers to viewing only their students records.

This can be achievable by simply applying an SQL “SELECT WHERE” statement to only grab students whose teacherID matches the teacher using the system.

### 6.3 - Teachers can submit lesson reports after each lesson.

As stated previously access rights are not implemented, so all users of the prototype are able to submit lesson reports. However this can be changed by restricting access to the submit button by any user other than teachers in the final product.

The functionality of submitting a report is implemented in the prototype:

Student Bookings						Lesson Report	Lesson Plan
ID	LessonPlansID	StudentID	TeacherID	attended	student_behav		
85	47	28	20	1	1		
86	23	44	34	1	2		
87	38	36	3	0	2		
88	32	8	21	0	1		
89	42	34	7	0	0		
90	44	48	26	1	2		
91	23	31	49	0	4		
92	2	11	18	1	1		
93	18	26	21	0	3		
94	34	25	48	0	2		
95	49	17	9	1	4		
96	19	22	20	0	0		
97	23	11	46	1	3		
98	11	3	5	1	1		
99	45	29	29	0	1		
100	19	25	26	1	4		

Record

ID: <input type="text" value="101"/>	LessonPlansID: <input type="text" value="32"/>	StudentID: <input type="text" value="7"/>
TeacherID: <input type="text" value="12"/>	attended: <input type="text" value="1"/>	student_behaviour: <input type="text" value="3"/>
notes: <input type="text"/>	date: <input type="text" value="05/05/2023"/>	

Commands

<input type="button" value="Clear Fields"/>	<input type="button" value="Search"/>	<input type="button" value="Update"/>	<input type="button" value="Add Record"/>	<input type="button" value="Delete Record"/>
---	---------------------------------------	---------------------------------------	---	--

Student Bookings						Lesson Report	Lesson Plan
ID	LessonPlansID	StudentID	TeacherID	attended	student_behav		
86	23	44	34	1	2		
87	38	36	3	0	2		
88	32	8	21	0	1		
89	42	34	7	0	0		
90	44	48	26	1	2		
91	23	31	49	0	4		
92	2	11	18	1	1		
93	18	26	21	0	3		
94	34	25	48	0	2		
95	49	17	9	1	4		
96	19	22	20	0	0		
97	23	11	46	1	3		
98	11	3	5	1	1		
99	45	29	29	0	1		
100	19	25	26	1	4		
101	32	7	12	1	3		

Record

ID:	<input type="text" value="101"/>	LessonPlansID:	<input type="text" value="32"/>	StudentID:	<input type="text" value="7"/>
TeacherID:	<input type="text" value="12"/>	attended:	<input type="text" value="1"/>	student_behaviour:	<input type="text" value="3"/>
notes:	<input type="text" value="None"/>	date:	<input type="text" value="05/05/2023"/>		

Commands

<a href="#">Clear Fields</a>	<a href="#">Search</a>	<a href="#">Update</a>	<a href="#">Add Record</a>	<a href="#">Delete Record</a>
------------------------------	------------------------	------------------------	----------------------------	-------------------------------

This is the same method of submitting a new record to all the other records the system manages.

The notes can be edited by using the frame on the right as it is more manageable than entering a large amount of text in a smaller entry field. The save button will simply apply the changes of the “notes” column to the record selected.

6.4-5 - Managers and teachers can create and view a searchable and sorted list of lesson plans.

Users can create and navigate lesson plans using the same methods as explained above.

ID	lessonTitle	lessonObjective	materials	procedure
1	Introduction to Piano Playing	Mastering complex drum patterns and rhythms	Textbook, music notation	Start by practicing basic rhythms and stick control on a practice
2	Introduction to Piano Playing	Learning basic chords and strumming patterns on the guitar	Instrument	
3	Advanced Drums and Percussion	Understanding music notation and basic theory concepts	Instrument	
4	Guitar Chords and Techniques	Mastering complex drum patterns and rhythms	Instrument	
5	Advanced Drums and Percussion	Developing proper hand and finger technique on the piano	Textbook	
6	Guitar Chords and Techniques	Understanding music notation and basic theory concepts	Piano	
7	Advanced Drums and Percussion	Improving vocal range and control for better singing performance	Instrument	
8	Guitar Chords and Techniques	Improving vocal range and control for better singing performance	Instrument	
9	Songwriting and Composition	Exploring different songwriting techniques and approaches	Instrument	
10	Guitar Chords and Techniques	Improving vocal range and control for better singing performance	Instrument	
11	Advanced Drums and Percussion	Learning basic chords and strumming patterns on the guitar	Textbook	
12	Songwriting and Composition	Improving vocal range and control for better singing performance	Instrument	
13	Introduction to Piano Playing	Mastering complex drum patterns and rhythms	Textbook	
14	Music Theory for Beginners	Exploring different songwriting techniques and approaches	Instrument	
15	Advanced Drums and Percussion	Exploring different songwriting techniques and approaches	Textbook	
16	Introduction to Piano Playing	Learning basic chords and strumming patterns on the guitar	Textbook	

**Record**

ID:  lessonTitle:  lessonObjective:  materials:  procedure:

**Commands**

[Clear Fields](#) [Search](#) [Update](#) [Add Record](#) [Delete Record](#)

## 7. Reporting and Data Analysis:

7.1 - Calculate and display sales report given a date range.

The prototype can display a sales report as demonstrated briefly in section 5.3. The report is calculated by selecting all the records the user wants to evaluate, and is displayed in the text frame on the right.

With this system the prototype has the ability to calculate revenue calculations for specific days by selecting records with the desired date range. The filter and sort feature of the program makes this much easier:

In this example we want to calculate the income from cafe sales in March 2022:  
the results can be sorted by dates:

The screenshot shows a software interface for managing sales data. At the top, there are two tabs: "Music Sales" and "Cafe Sales". The "Cafe Sales" tab is selected, displaying a grid of sales records. The columns in the grid are: ID, CafeltemID, CustomerID, count, and date. The data in the grid is as follows:

ID	CafeltemID	CustomerID	count	date
1166	22	1	1	01/04/2022
643	30	5	3	24/03/2022
208	43	39	6	24/03/2022
449	39	17	3	22/03/2022
612	10	5	7	20/03/2022
1124	4	37	5	19/03/2022
301	21	33	3	19/03/2022
1112	47	41	10	18/03/2022
1083	26	42	3	14/03/2022
340	29	1	6	12/03/2022
309	14	17	9	07/03/2022
406	6	2	3	04/03/2022
565	30	38	7	02/03/2022
542	29	24	1	27/02/2022
1040	33	37	5	26/02/2022
884	13	40	6	26/02/2022

To the right of the grid, a summary panel displays "Sales Estimate" with the total revenue listed as £577.86. Below this, there is a section for "Sale Commands" with buttons for "Edit" and "Save Changes".

Below the grid, there is a "Record" section containing input fields for ID, CafeltemID, CustomerID, count, and date. The values shown are ID: 643, CafeltemID: 30, CustomerID: 5, count: 3, and date: 24/03/2022.

At the bottom, there is a "Commands" section with buttons for "Clear Fields", "Search", "Update", "Add Record", and "Delete Record".

With the results sorted by date the user can simply click on the first day in March and shift+click the last date, selecting them all. The system will automatically calculate the results from the selected records and display them in the text frame on the right.

### 7.2 - Calculate and display value of inventory.

The method described in **7.1** can be implemented similarly in the inventory system to calculate stock between certain dates. From this the user can calculate the inventory value at certain dates or date ranges.

### 7.3 - Display estimate of revenue from student lessons.

In the “Student Bookings” tab the user can select a range of lessons and estimate the monthly revenue from these values. The estimate is calculated from looking at each lesson’s frequency (weekly, fortnightly, or monthly), lesson cost, and whether or not the lesson was cancelled. The results from the calculation are formatted and displayed in the text frame. The estimate can then be exported to a text file:

The screenshot shows a software application window with three tabs: "Student Bookings", "Lesson Report", and "Lesson Plan". The "Student Bookings" tab is active, displaying a table of 16 student bookings with columns for ID, StudentID, TeacherID, location, lesson\_frequency, and lesson\_day. The "Lesson Report" tab is visible on the right, containing text output related to user IDs and their lesson types. The "Lesson Plan" tab is also visible. At the bottom, there is a "Record" section with input fields for ID, StudentID, TeacherID, location, lesson\_frequency, and lesson\_day, along with an "Export" section containing "Calculate", "Clear", and "Export as .txt file" buttons.

ID	StudentID	TeacherID	location	lesson_frequency	lesson_day
1	93933250	97904097	Room 10	Fortnightly	Wednesday
2	23078257	20588786	Room 5	Monthly	Friday
3	35794237	74234495	Room 9	Weekly	Tuesday
4	37601708	23868595	Room 5	Fortnightly	Friday
5	42880367	30548433	Room 6	Fortnightly	Monday
6	58206941	51551392	Room 10	Fortnightly	Monday
7	36918344	23862463	Room 7	Fortnightly	Thursday
8	55479729	36566107	Room 10	Monthly	Wednesday
9	34505342	811151522	Room 9	Fortnightly	Monday
10	89365844	49711713	Room 9	Fortnightly	Monday
11	57739386	49852907	Room 4	Weekly	Wednesday
12	11488486	35765230	Room 9	Monthly	Tuesday
13	83471402	55583555	Room 8	Weekly	Thursday
14	76380853	98492717	Room 9	Monthly	Monday
15	65557280	20327883	Room 10	Monthly	Wednesday
16	39544820	97437566	Room 2	Weekly	Monday

User ID '4' has Fortnightly monthly lessons, each costing £20 resulting in a monthly revenue of: £40.0  
User ID '5' has Fortnightly monthly lessons, each costing £10 resulting in a monthly revenue of: £20.0  
User ID '6' has Fortnightly monthly lessons, each costing £10 resulting in a monthly revenue of: £20.0  
  
Total Monthly Estimate for Selected Lessons: £80.0

Record

ID:	4	StudentID:	37601708	TeacherID:	23868595
location:	Room 5	lesson_frequency:	Fortnightly	lesson_day:	Friday

Export

Calculate	Clear
Export as .txt file	

Here the calculate button was pressed and we can see that of these 4 bookings in the system 1 of them was cancelled (it is not being displayed in the report) and the other 3 total a monthly revenue of £80

The export button can be used to append the calculation to a report file:

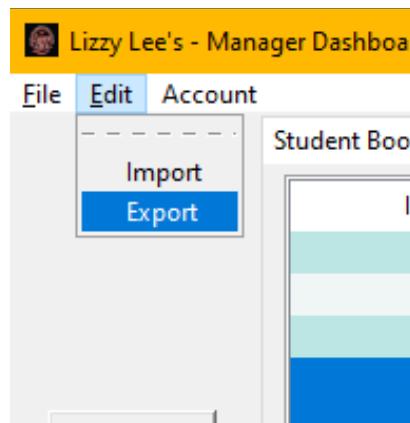
```

LessonEstimate.txt
1 =====
2 06/05/2023: STUDENT_BOOKING Monthly Revenue Estimate:
3 User ID '2' has Monthly monthly lessons, each costing £15 resulting in a monthly revenue of: £15.0
4 User ID '3' has Weekly monthly lessons, each costing £20 resulting in a monthly revenue of: £80.0
5 User ID '4' has Fortnightly monthly lessons, each costing £20 resulting in a monthly revenue of: £40.0
6 User ID '5' has Fortnightly monthly lessons, each costing £10 resulting in a monthly revenue of: £20.0
7 User ID '6' has Fortnightly monthly lessons, each costing £10 resulting in a monthly revenue of: £20.0
8
9 Total Monthly Estimate for Selected Lessons: £175.0
10 =====
11 07/05/2023: STUDENT_BOOKING Monthly Revenue Estimate:
12 User ID '4' has Fortnightly monthly lessons, each costing £20 resulting in a monthly revenue of: £40.0
13 User ID '5' has Fortnightly monthly lessons, each costing £10 resulting in a monthly revenue of: £20.0
14 User ID '6' has Fortnightly monthly lessons, each costing £10 resulting in a monthly revenue of: £20.0
15
16 Total Monthly Estimate for Selected Lessons: £80.0
17

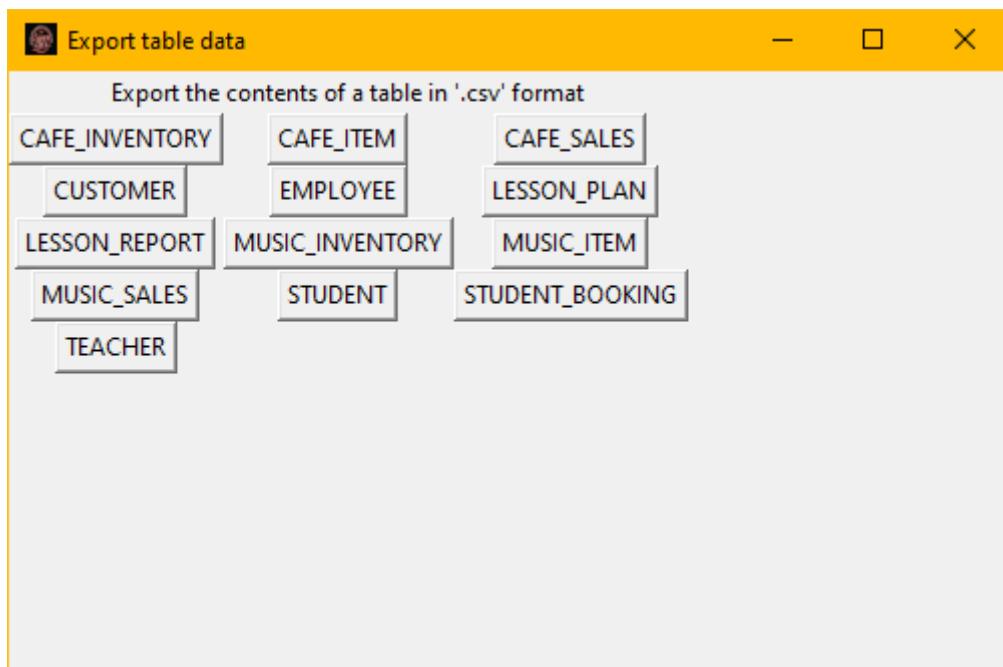
```

*7.4 - Support exporting calculations to .txt or .csv format.*

In addition to the export feature demonstrated in section **7.3** the prototype can also export entire tables in “.csv” format from the “Export” window accessible in the toolbar..



The rather underwhelming export window allows the user to select a table to export to a “.csv” file in the root folder of the application:



The results are a file named after the table containing the header information and the columns for each record:

	CafeItemID, CustomerID, count, date
1	35,39,9,27/10/2022
2	6,24,3,09/08/2019
3	9,23,3,22/11/2022
4	12,4,2,24/05/2019
5	24,33,7,18/03/2021
6	50,41,4,14/04/2021
7	36,34,4,26/09/2019
8	43,39,6,24/03/2022 ...
9	49,22,1,02/09/2019
995	32,8,4,10/01/2019
996	13,2,4,23/07/2021
997	39,38,1,16/01/2022
998	40,22,5,15/12/2019
999	25,20,4,26/01/2019
1000	19,10,7,19/04/2019
1001	

With this the user can export table data to be used in third party tools for data analysis beyond the scope of this project, or for transmission/storage of a specific table.

## 8. Data Security and Integrity

8.1 - Encrypt all stored data to prevent personal information being leaked.

All data is stored in an encrypted file using XOR key encryption.

An earlier version of the prototype made use of asymmetric encryption as was planned in the Design document, however, in practical use I discovered this to be extremely slow even for small amounts of data. Additionally, it came to my attention that there was no need for asymmetric encryption as all the files remain on the same device. In its place I opted for the much quicker symmetric XOR option.

```
encrypt(self, message):
    #Convert the message and key to byte arrays
    message_bytes = message.encode('utf-8')
    key_bytes = self.key.encode('utf-8')

    #XOR each byte of the message with the corresponding byte of the key
    encrypted_bytes = bytes([message_bytes[i] ^ key_bytes[i % len(key_bytes)] for i in range(len(message_bytes))])

    #Convert and return encrypted bytes to a string
    encrypted_message = encrypted_bytes.hex()
    return encrypted_message
```

To secure the system's user's privacy all password fields are always hidden. The password column is not displayed in the treeview frame and password entries are masked with asterisks.

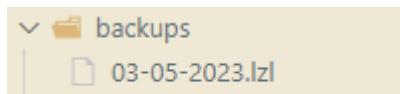
Record			
ID:	<input type="text" value="1"/>	username:	<input type="text" value="lwalters"/>
forename:	<input type="text" value="Llion"/>	surname:	<input type="text" value="Walters"/>
title:	<input type="text" value="Dr"/>	birthdate:	<input type="text" value="21/04/1981"/>
job_description:	<input type="text" value="Crew"/>	town:	<input type="text" value="Llanerchymedd"/>
postcode:	<input type="text" value="LL77 9ZI"/>	phone_num:	<input type="text" value="None"/>
		email:	<input type="text" value="None"/>
		password:	<input type="text" value="*****"/>

The password can still be updated but the user will not be able to see the original password, and only the manager can edit other user passwords, so users other than managers must still know their own password to change it.

#### 8.2 - Make routine backups of all stored data.

When the program is closed the system will create a new backup with the current date in the backup folder. With this method the system will create a backup for every new day the system is used.

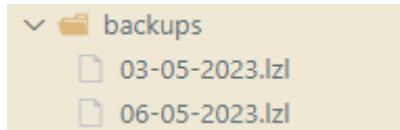
In this example the current date is 06/05/2023. The last backup was made on 03/05/2023



When the application is closed the backup file is saved using the current date.

Console Log:

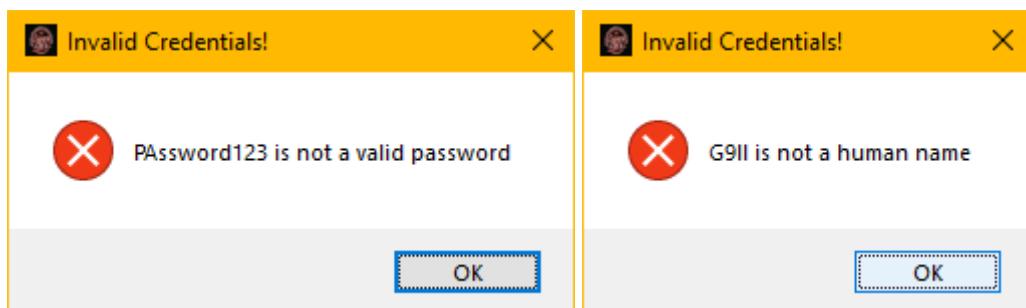
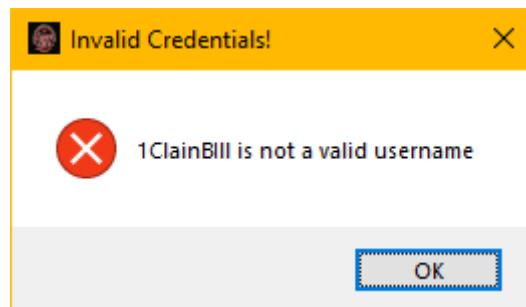
```
Backup created: backups/06-05-2023.lz1
Saved Encrypted File as: backups/06-05-2023.lz1
Saved Encrypted File as: savedata.lz1
App Closing
```

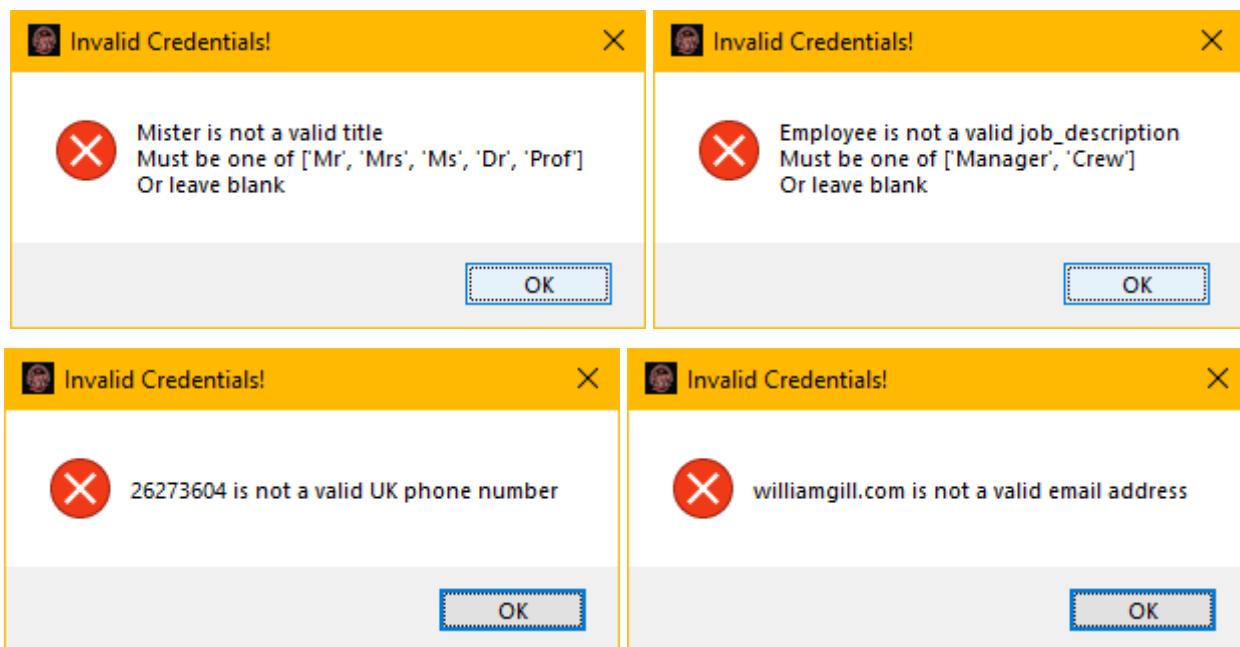


## 8.3 - Validate all user input.

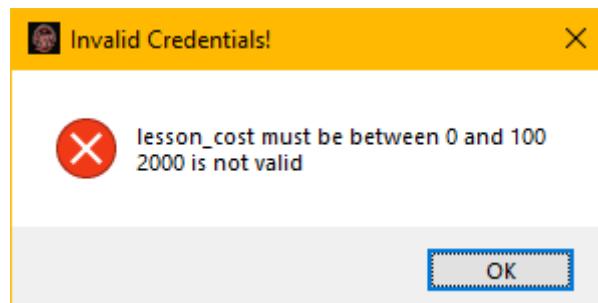
When data is entered in an entry field the system makes a check depending on the data type. With the following entries the system returns these errors:

Record					
ID:	<input type="text" value="1"/>	username:	<input type="text" value="1ClainBIII"/>	password:	<input type="text" value="*****"/>
forename:	<input type="text" value="William"/>	surname:	<input type="text" value="G9II"/>	sex:	<input type="text" value="F"/>
title:	<input type="text" value="Mister"/>	birthdate:	<input type="text" value="21/13/2020"/>	hire_date:	<input type="text" value="22/06/2022"/>
job_description:	<input type="text" value="Employee"/>	town:	<input type="text" value="Llanerchymedd"/>	county:	<input type="text" value="Gwynedd"/>
postcode:	<input type="text" value="LL77 9ZI"/>	phone_num:	<input type="text" value="26273604"/>	email:	<input type="text" value="williamgill.com"/>





Furthermore the system supports range checks. When inputting a lesson cost the system will reject anything outside the range of 0-100:



When the data is successfully validated the system will notice the user:



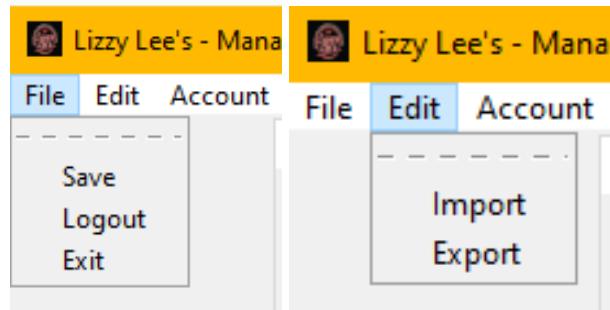
# Prototype Evaluation

The prototype successfully demonstrated how the system is able to achieve many of the objectives required by the stakeholders of Lizzy Lee's Rock Cafe. Of the 26 objectives set in the specification only 5 were unmet or needed altering - 4 of which can be solved in the same solution of handling user access. Needless to say there were more issues discovered during the production of the prototype that need to be addressed.

I believe the prototype demonstrated some crucial features of the final product and proved how flexible the software design is. By implementing classes new features could be added with ease, and further features still can be added built off the existing classes.

Some additional features or changes of the program that may not have been explicitly mentioned in the specification but were included in the prototype are:

**Toolbar:**

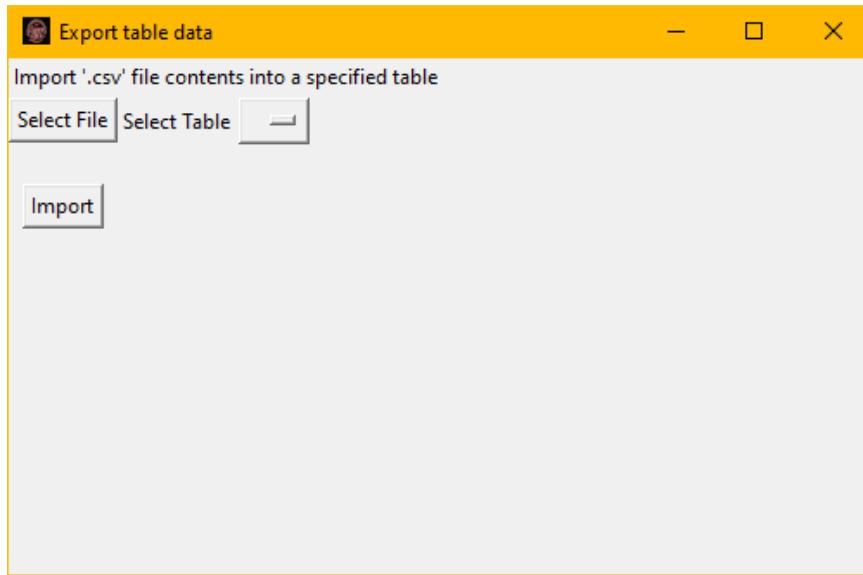


“Save” saves the file without closing the application

“Logout” saves the file and returns to the login screen

“Exit” saves the file, a backup, and exits the application

“Import” opens an import window allowing the user to select a “.csv” file and table to import data to:



“Export” opens the export window demonstrated in section **7.4**

#### Encryption:

When developing the prototype I discovered that the asymmetric encryption was not appropriate for this project. The load times were exceeding 20 seconds for only 50 records. As explained in section **8.1** I ended up opting for XOR encryption which is symmetric and significantly faster, whilst maintaining a high level of security.

The new system is able to comfortably load thousands of records near instantaneously. The details of this encryption method are explained in **8.1**

## REVIEW:

### Levels of Access:

The prototype has fallen short in a few areas. First and foremost, the user access levels not being properly integrated is an issue. Many aspects of the system will change depending on the user's access level. To fix this I need to add an attribute to classes that will be used to disable or restrict the class depending on the user's access level.

### Data Readability:

Another issue is the layout of data. In some tabs the display is sufficient - such as the user records where all the record fields are self-identifying by their column names; "username", "forename", "hire\_date" etc., but in some tabs like the "Lesson Report" tab the data presented in the treeview frame is highly unreadable by humans:

ID	LessonPlansID	StudentID	TeacherID	attended	student_behavior
1	50	4	32	0	1
2	35	32	48	0	2
3	45	22	24	1	0
4	37	3	40	0	0
5	49	26	33	1	0
6	33	25	4	1	0
7	34	35	46	1	2
8	42	45	31	1	1
9	29	3	50	0	0
10	49	5	27	0	4
11	5	21	21	0	0
12	15	25	17	1	3
13	41	10	27	0	2
14	10	47	35	0	0
15	11	12	32	1	1
16	19	46	3	1	3

The results are mostly meaningless numbers referencing other SQL tables. This could be made more intuitive and useful by replacing the displayed data to represent the full name of the students and teachers in their ID columns, and replacing the LessonPlansIDs with lesson titles.

Furthermore the "attended" field could be parsed as yes/no instead of 1/0 but internally maintain storing the result as 1/0 to reduce storage space and reading times.

**Adding Records:**

When adding records the system requires the user to input an ID, but this is time consuming and pointless. The user should not have to manually input an ID that has not already been used - especially because the ID is a computer defined field used exclusively by and for the computer - but instead the computer should automatically create the record with the next ID freeing up an input from the human and potential error from human input.

Additionally, spinboxes were not implemented. These spinboxes could have made lookup check fields more intuitive for users, and speed up the process of adding records to the system.

**Importing Data:**

When importing data the system does not execute validation checks. This is a very simple design flaw that can be very detrimental to the system if it is exploited. This can be addressed by validating fields as they are imported.

**Error Handling:**

The system does not fully make use of error handling and system warnings. When importing or exporting data there is no indication to the end user if the import/export was successful or not, and if unsuccessful it should display why.

**Style:**

There is also the consideration of styling the application to be more attractive to users. A good looking design can boost productivity and help with the intuitive feel of a program. To change this I could use the Tkinter themes or create my own theme to match Lizzy Lee's aesthetic.

# Post-Prototype Design

## STAKEHOLDER REPORT:

After completing the prototype I have acquired feedback from the primary stakeholder of the system; Liz Pryde - the manager of Lizzy Lee's. She was provided with the application and asked to produce a report listing any changes she may require of the program before the final product is designed.

I asked her to spend some time with the program and produce the report of the system and this is her response taken from an email (verbatim):

**Thank you William for taking the time to create this application for our company. Overall, I think it has great potential, but there are a few areas that could be improved:**

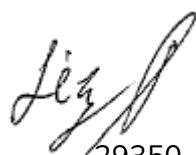
**The program should look nicer. While functionality is important, it would be great if the program also had a visually appealing and intuitive interface. A visually pleasing design can make it easier for employees to use and navigate the system.**

**The process of adding records should be more intuitive and less hassling. Currently, the requirement to input the correct ID to add a record can be a hassle for our employees. It would be great if we could simplify this process and make it more intuitive, perhaps by using a more user-friendly interface or by eliminating the need to input IDs altogether.**

**There should be an option to filter out results in a specific date, or price range: Being able to filter results by a range is an important feature that would allow us to better analyse history of data. It would be great if this feature could be added to the system.**

**Also, I noticed the "Account" option in the toolbar doesn't seem to do anything. I thought it might be an issue with my computer but it won't work with any computers I have used.**

**Finally, can I ask that you add a feature to export the sales and inventory calculations like you can do with music lesson revenue estimates? This would make recording the history of transactions easier.**



Liz  
29350

William C. Gill

## REPORT EVALUATION:

Liz Pryde's main concern seems to be the design of the application. The prototype does not have proper emphasis on the purpose of each widget and can be confusing to navigate. Whilst most features of the specification have been implemented in the prototype it is not readily apparent how to access these features such as the import/export features, or exporting files in the "Lesson Booking" tab.

She also mentions some issues with adding records to the system's database. This concern of hers matches what I had evaluated in my prototype assessment. Liz Pryde has requested that the ID field be made more intuitive or simply removed. From this I believe the best approach would be to remove user input from the ID field, and any foreign keys should use a system-assisted method of filling in the correct ID such as displaying a list of possible results to the user.

Although not explicitly mentioned in the report another fix that needs to be adjusted in the process of adding records is implementing a Tkinter Listbox feature for entry fields that perform lookup checks. A Listbox will force the user to select a viable option, will speed up inputting data by selecting from an option instead of typing in the field, and will also help users understand what data the entry field is looking to accept.

Liz Pryde's suggestion to filter results by range is an excellent idea that can be implemented to further improve the process of filing large amounts of data. This can be achieved by adding a "range check" frame by the "entries" frame to allow the user to filter the treeview results.

The "Account" button Liz Pryde mentions here was planned but never given a purpose. The toolbar button was intended to contain the logout function, but this was instead included in the "File" toolbar option. "Account" will be removed as it has no use.

Finally Liz Pryde has asked that the final product should allow its users to export sales and inventory calculations in a similar fashion to how the user can perform and export calculations in the "lesson bookings" tab. This can easily be implemented as the skeleton of the function is already implemented, and only has to be migrated to specifically work for the "Sales" and "Inventory" tabs.

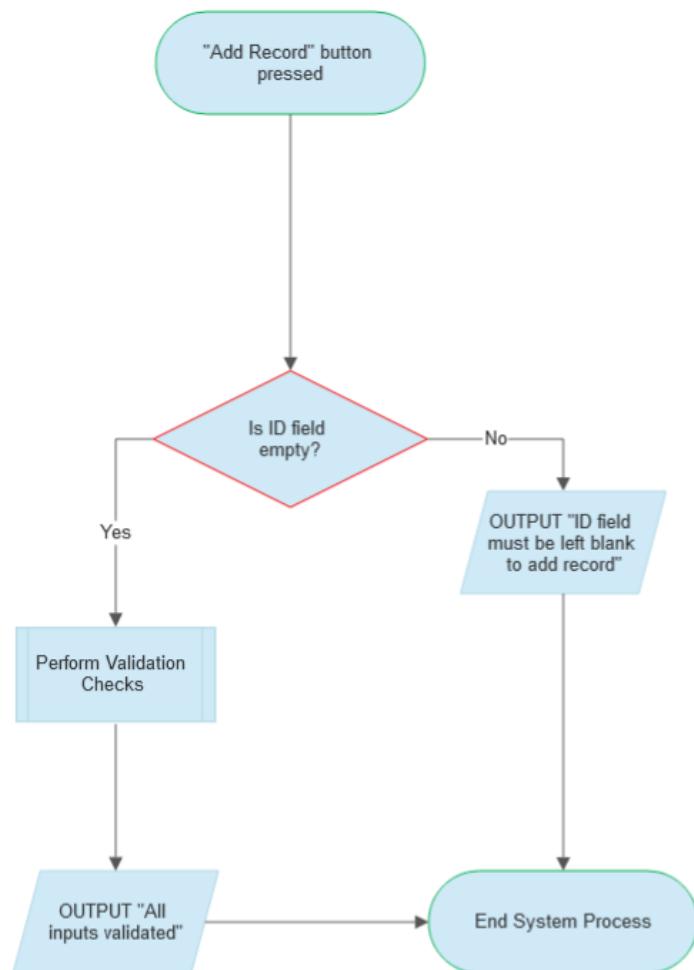
# REVISED DESIGN

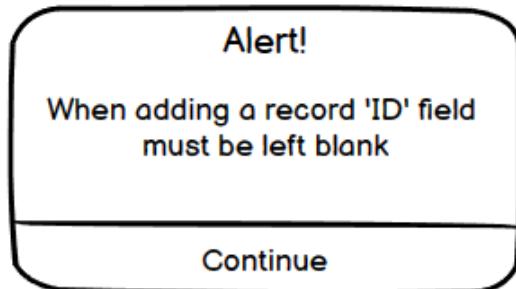
In response to Liz Pryde's feedback this is what I have determined needs to be implemented or changed in the final product:

## Adding Records:

I agree that the process of adding a record to the system's database could be improved. As addressed in the Feedback document and Liz Pryde's report there should be no requirement to input an ID for adding a record, and additionally I have determined that adding a foreign key should be made easier - such as giving a list of options to choose from.

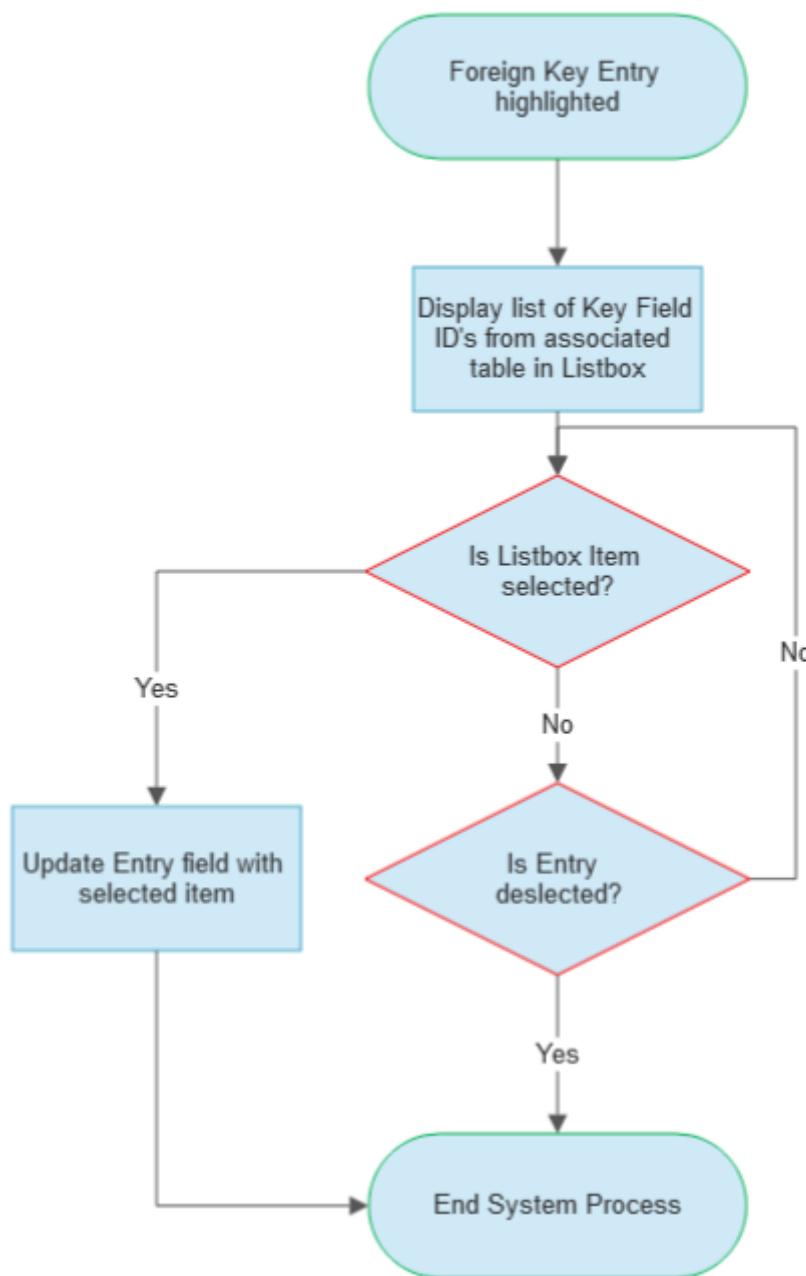
To fix the issue of requiring an ID to add a record I can simply add a check to ensure the ID field is blank when the "Add Record" button is pressed. If the ID is not empty a system prompt will warn the user that ID must be left blank to add an ID:





Requiring ID to be left blank will allow the system to automatically decide the ID value to be used by adding 1 to the last used ID.

Regarding using foreign keys this can be made easier by displaying a list of possible foreign key values to the user:



Creating a list of possible foreign keys will be achievable by creating a tkinter Listbox underneath the entry widgets when the entry is selected. The listbox will be filled with key fields from the associated table, and if one is selected the listbox will be destroyed and the entry is updated.

## Filtering By Range:

In addition to filtering results by the inputs of the entry fields there should be a method of filtering results between a given range for fields such as “Dates” and price options.

ID	Name	Job Title	Title
1	Giacomo Guilizzoni	Founder & CEO	Mr
2	Marco Botton	Tuttofare	Dr
3	Mariah MacLachlan	Better Half	Mrs
4	Valerie Liberty	Head Chef	Mr

The program will automatically execute filters on the list of data in the table selecting only dates that fit within the range of results grabbed. The system will work the same with dates.

## Exporting sales and inventory estimates:

Liz Pryde also suggested the system should be able to export the calculations made in the “Sales” and “Inventory” tabs. This would be an easy addition as it has already been integrated in the “Lesson Bookings” tab and it would benefit the stakeholders.

Some changes can be made to improve the readability of the results, such as logging the date of the estimate, and the items used to calculate the estimate.

## Design:

Another change that needs to be made is the design of the application.

### Tooltips:

To help users with understanding a widget or function's use tooltips should be included for processes that explain to the user how they are used and if relevant their use case.

This can be done by creating a Tooltip class in python that takes a widget as a parameter and displays the given text when the widget is hovered over for a period of time.

The parameters of the class that will be passed into attributes: **widget, text, hoverTime**

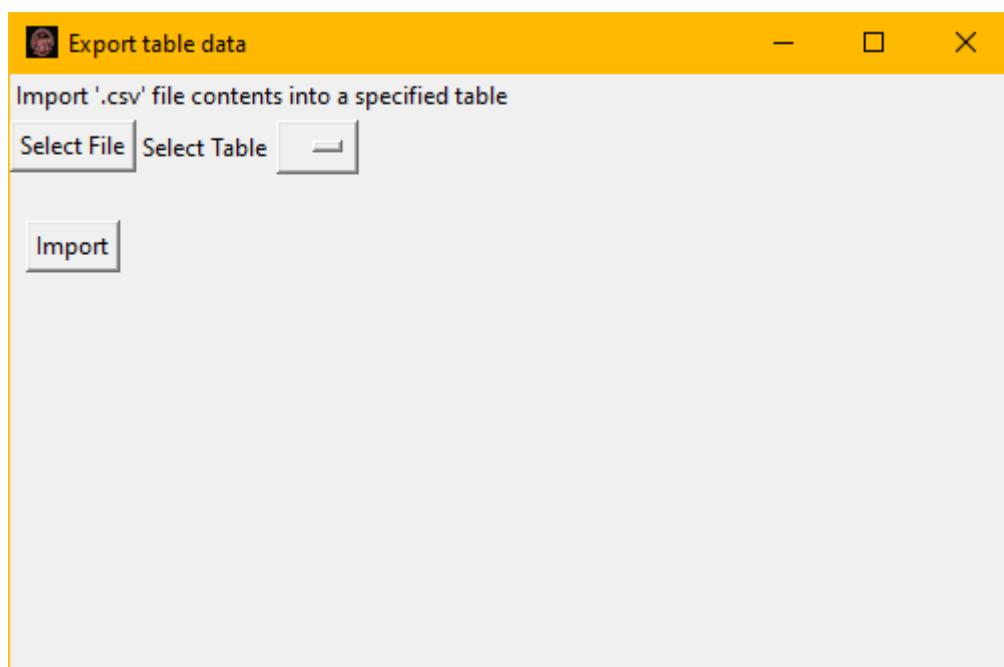
Methods of the class will be: **showTooltip()** - creates a Tkinter Toplevel at the mouse cursor that contains the text in the text attribute, **hideTooltip()** - destroys the created Toplevel

### Window Layout:

Some windows such as the “Export” and “Import” windows are extremely cluttered and lack feedback to the user regarding their inputs.

These windows specifically need to have a rehaul in design to match the layout specified in the Design document:

#### Current Import screen:



Designed Import Screen:

Lizzy Lee's | Batch Import

## Import From CSV ?

Select Table

Employee ▼

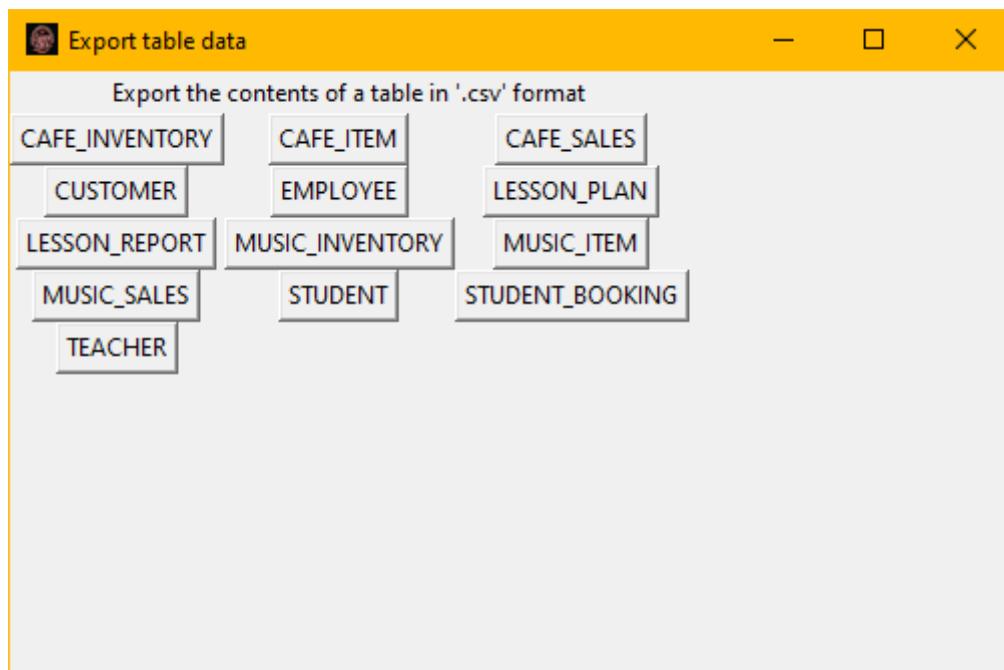
Teacher  
Student  
Customer  
Inventory

Imports the contents of a .csv file into the selected table.  
Make sure the CSV file follows the format of the selected table and obeys its validation rules!

Upload a file:

Browse...

Batch Import

Current Export Screen:

Designed Export Screen:

Lizzy Lee's | Export

## Export Data ②

Select Table

Employee ▾

Teacher  
Student  
Customer  
Inventory

Exports the contents of the selected table to a .csv file

Select a file:

Browse...

Batch Import

Additionally, other aspects of the program will need various touches:

**Treeview heading styling:**

The headers to columns in every table should be stylised to match natural human formats.

Words should begin capitalised and underscores ('\_') should be replaced with spaces:

birthdate	hire_date	job_description
21/04/1981	22/06/2022	Crew
05/04/1984	06/12/2018	Manager
02/12/1969	21/12/2018	Crew
16/12/1967	06/02/2019	Manager
13/10/1973	11/10/2015	Manager
25/08/1994	16/09/2019	Manager

birthdate -> Birthdate

hire\_date -> Hire Date

job\_description -> Job Description

**Entry label styling:**

similar to styling column headers the entry labels should be styled using the same rules:

Record

ID:	<input type="text"/>	username:	<input type="text"/>	password:	<input type="text"/>
forename:	<input type="text"/>	surname:	<input type="text"/>	sex:	<input type="text"/>
title:	<input type="text"/>	birthdate:	<input type="text"/>	hire_date:	<input type="text"/>
job_description:	<input type="text"/>	town:	<input type="text"/>	county:	<input type="text"/>
postcode:	<input type="text"/>	phone_num:	<input type="text"/>	email:	<input type="text"/>

**Button Padding:**

The command buttons under the records can be padded and more evenly spaced to look more appealing to the user

Commands

<a href="#">Clear Fields</a>	<a href="#">Search</a>	<a href="#">Update</a>	<a href="#">Add Record</a>	<a href="#">Delete Record</a>
------------------------------	------------------------	------------------------	----------------------------	-------------------------------

**Automatic Column Sizing:**

The column headers can be made larger for columns that contain more text. This is especially apparent in the “Lesson Plan” tab that contains text descriptions which usually extend beyond the default width of columns:

Student Bookings | Lesson Report | Lesson Plan

ID	lessonTitle	lessonObjective	materials	procedure
1	Introduction to Piano	Mastering complex chords	Textbook, music notes	Start by practicing basic chords
2	Introduction to Piano	Learning basic chord progression	Instrument of choice, sheet music	Begin by reviewing basic chords
3	Advanced Drums and Percussion	Understanding music theory	Instrument of choice, online resources	Review the basics of rhythm and timing
4	Guitar Chords and Techniques	Mastering complex chords	Instrument of choice, sheet music	Start by learning basic chords and scales
5	Advanced Drums and Percussion	Developing proper hand technique	Textbook, music notes	Begin by reviewing basic drumming techniques
6	Guitar Chords and Techniques	Understanding music theory	Piano keyboard, sheet music	Start by practicing basic chords and scales
7	Advanced Drums and Percussion	Improving vocal range	Instrument of choice, online resources	Explore different apps for vocal training
8	Guitar Chords and Techniques	Improving vocal range	Instrument of choice, sheet music	Start by learning basic vocal exercises
9	Songwriting and Composition	Exploring different song structures	Instrument of choice, online resources	Explore different app for songwriting
10	Guitar Chords and Techniques	Improving vocal range	Guitar, pick, music stand	Begin by warming up the vocal cords
11	Advanced Drums and Percussion	Learning basic chord progression	Textbook, music notes	Review the basics of rhythm and timing
12	Songwriting and Composition	Improving vocal range	Drum kit, drumsticks	Start by learning basic drumming techniques
13	Introduction to Piano	Mastering complex chords	Drum kit, drumsticks	Explore different apps for drumming
14	Music Theory for Beginners	Exploring different song structures	Drum kit, drumsticks	Start by practicing basic music theory concepts
15	Advanced Drums and Percussion	Exploring different song structures	Textbook, music notes	Review the basics of rhythm and timing
16	Introduction to Piano	Learning basic chord progression	Guitar, pick, music stand	Review the basics of guitar chords and scales

The user does have the ability to manually resize columns to fit the data in, but this should not be required when resizing is obviously necessary such as in this instance.

### Formatting Treeview Results:

As mentioned in my assessment of the prototype in the Prototype document I determined that the treeview results should be formatted for easier user comprehension of data. Specifically, the program should replace foreign key identifiers with more meaningful data such as a teacher's full name replacing their foreign key in the "Lesson Report" table.

ID	LessonPlansID	StudentID	TeacherID	attended	student_behav
1	50	4	32	0	1
2	35	32	48	0	2
3	45	22	24	1	0
4	37	3	40	0	0
5	49	26	33	1	0
6	33	25	4	1	0
7	34	35	46	1	2
8	42	45	31	1	1
9	29	3	50	0	0
10	49	5	27	0	4
11	5	21	21	0	0
12	15	25	17	1	3
13	41	10	27	0	2
14	10	47	35	0	0
15	11	12	32	1	1
16	19	46	3	1	3

Foreign key fields should be replaced with a meaningful identifier in the treeview results but maintain the actual value of the key field internally so that the system will have no mistake when identifying the record of the associated table.

With these rules the treeview should return something like this:

Lesson Bookings				
User Records	ID	Student	Teacher	Price
Lessons	1	Evie Jones	Jane Doe	65
Stock	2	Mared Hughes	Harold Dickinson	65
Sales	3	Tomos Edwards	Jane Doe	55

ID	1	Price	65	<input type="button" value=""/>	<input type="button" value=""/>
StudentID	12	<input type="button" value=""/>	<input type="button" value=""/>	<input type="button" value=""/>	<input type="button" value=""/>
TeacherID	3	<input type="button" value=""/>	<input type="button" value=""/>	<input type="button" value=""/>	<input type="button" value=""/>

When the user selects a record it will still return the actual data stored in the record instead of grabbing arbitrary students' names which could be duplicate across several student records.

## System Messages:

In various areas of the prototype the system does not respond to the user responding to their input. This is most apparent in the “Export” and “Import” windows, which have no indication of whether or not the data was successfully imported/exported. This can be changed by implementing the message box alerts that have been used in the validation system.

When the user attempts to import or export data the system could use the Python “Try/Except” feature to account for errors in the process, and warn the user if something went wrong. If there were no errors the system can inform the user that the import/export was successful.

## **Additional Changes:**

Additional changes that need to be made are:

### **Validation checks when importing data:**

The prototype does not check for valid inputs when importing data from a “.csv” file. This has led to some issues already when developing the prototype, and if left unchanged could lead to significant issues with using the application in production such as ignoring incorrectly formatted dates from search queries and filters.

This can be solved by applying the same validation rules to each record added to the system from an import and warning the user if the file has invalid data.

# REVISED SPECIFICATION

From Liz Pryde's report and my own report made after the prototype I have revised the list of features that need to be included in the final product:

**Red Bold** objectives are new to the specification from the prototype.

**Yellow Italics** objectives need to be completed from the prototype.

## 1. User Access and Login System:

1.1 - Restrict app to allow different levels of access for managers, employees, teachers, and students.

1.2 - Login system to authenticate users and grant appropriate permissions. *Users will only have access to the features of the GUI their access levels allow.*

## 2. Customer Records:

2.1 - Managers can create, update, and delete validated customer records with their personal information.

2.2 - Display list of customers with personal information to managers in a searchable and sorted table.

2.3 - Display the purchase history of customers in a searchable and sorted table.

**2.4 - Filter records from a range of birthdates**

## 3. Employee Records:

3.1 - Managers can create, update, and delete validated employee records with their personal information.

3.2 - Display list of employees with personal information to managers in a searchable and sorted table.

3.3 - *Employees can edit their own records.*

**3.4 - Filter records from a range of birthdates and hire dates**

## 4. Teacher and Student Records:

4.1 - Managers can create, update, and delete validated teacher and student profiles with their personal information.

4.2 - Display list of teachers and students with personal information to managers in a searchable and sorted table.

4.3 - *Teachers and students can edit their own records*

4.4 - *Teachers* and managers can view the progress and attendance of students in their respective lessons.

**4.5 - Filter records from a range of birthdates and hire dates (for teachers)**

**5. Sales and Inventory Management:**

- 5.1 - Managers can create, update, and delete validated stock items for both the music store and the cafe with detailed information, such as product name, description, and price.
- 5.2 - Record inventory levels over a period of time.
- 5.3 - Track sales from user input.
- 5.4 - ***Automatically update inventory levels with sales.***
- 5.5 - Allow managers to add items to inventory levels.
- 5.6 - Filter records from a range of date options and price options**

**6. Student Lessons:**

- 6.1 - Managers can create, update, and delete each student's lesson bookings.
- 6.2 - Managers and **teachers** can view searchable and sorted lists of student reports.
- 6.3 - **Teachers** can submit lesson reports after each lesson.
- 6.4 - Managers can create lesson plans.
- 6.5 - Managers and teachers can view a searchable and sorted list of lesson plans.
- 6.6 - Filter records from a range of lesson dates and costs**

**7. Reporting and Data Analysis:**

- 7.1 - Calculate and display sales report ***given a date range.***
- 7.2 - Calculate and display value of inventory.
- 7.3 - Display estimate of revenue from student lessons.
- 7.4 - Support exporting calculations to .txt or .csv format.

**8. Data Security and Integrity**

- 8.1 - Encrypt all stored data to prevent personal information being leaked.
- 8.2 - Make routine backups of all stored data.
- 8.3 - Validate all user input.

# Software Development

## SETUP

The final digital solution for Lizzy Lee's Rock Cafe has been developed and is fully functional.

To use the system the user needs the “main.py” file alone, however to demonstrate it’s abilities I have included a “savedata.lzl” file that has been filled with thousands of records already.

If the program is run without an existing save file already made it will create a default admin account with manager access levels. The system will tell the user its credentials until they first login.

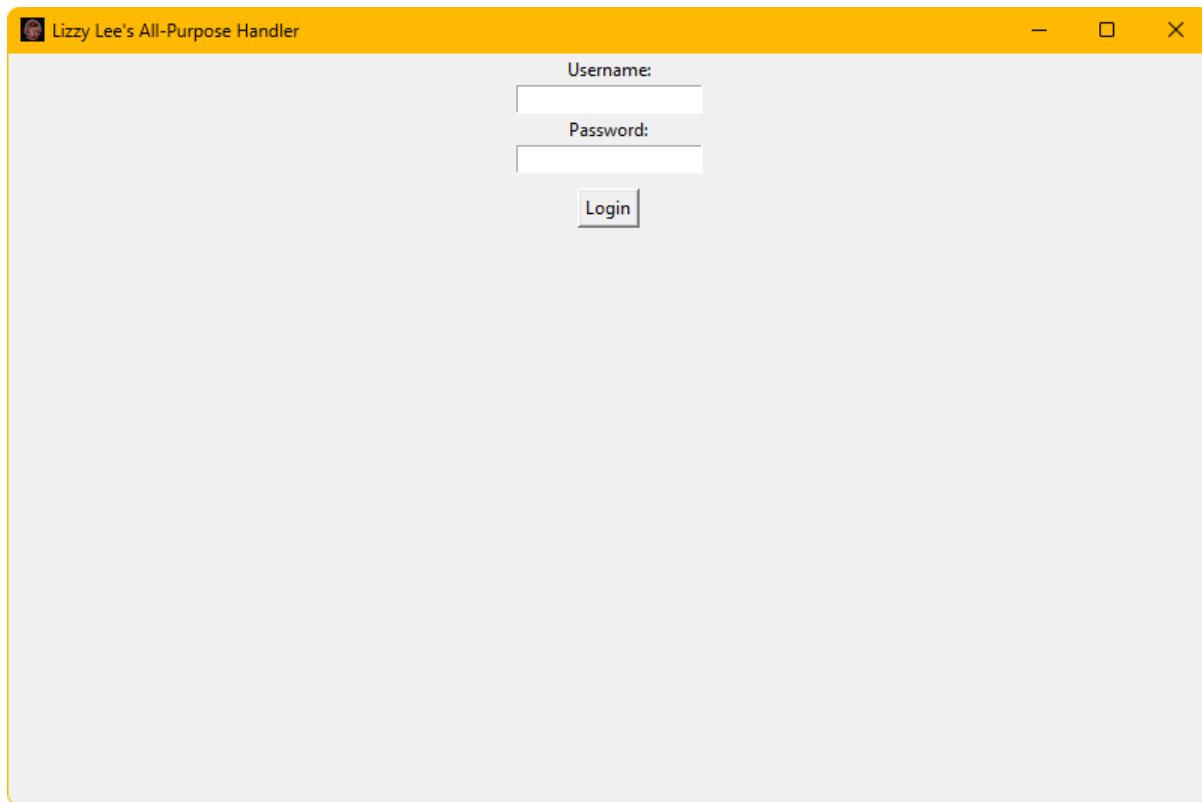
When the program is closed it will automatically make a backup file with the day’s date as its name in a backup folder.

A “Lizzy Lees Logo.ico” is also used by the system to display the company logo on all of the windows.

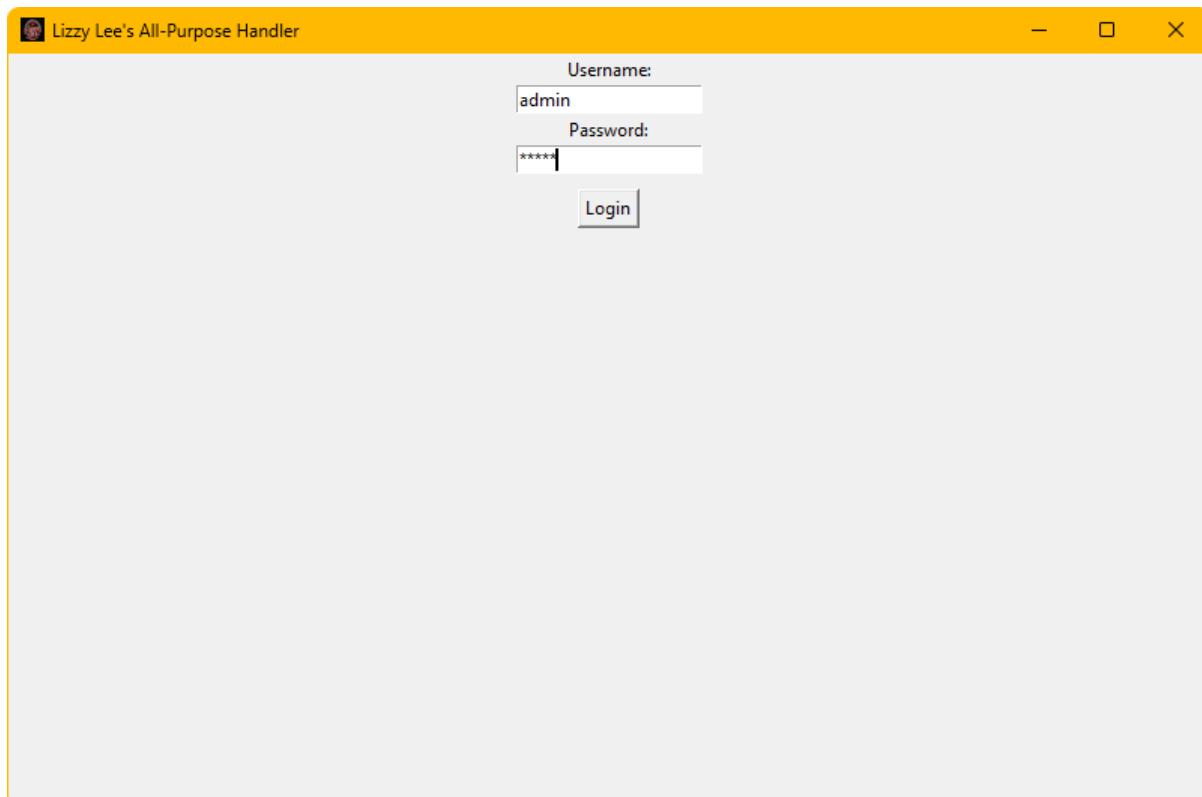
The default admin account has been left in the example database. Along with it are the following records that can be logged into to explore the full use case of the program

Access Level	Manager	Employee	Teacher	Student
username	admin	employee	teacher	student
Password	admin	Passw0rd!	Passw0rd!	Passw0rd!

The login screen here takes a Username and Password input. Any of the credentials above will allow the user access to the system - along with the other credentials already in the system.



Using 'admin' as an example:



The “Manager Dashboard” looks like this:

ID	Username	Forename	Surname	Sex	Title	Birthdate	Hire Date	Job Description	Town
1	admin	None	None	None	None	None	None	Manager	None
2	employee	None	None	None	None	None	None	Employee	None
3	manager	Nia	Fychan	M	Dr	07/12/2000	02/06/2012	Manager	Newborough
4	alloyd	Aled	Lloyd	F	Ms	12/12/1961	27/02/2019	Employee	Rhosneigr
5	ndavies	Nia	Davies	F	Dr	18/07/1981	13/11/2012	Manager	Llanddona
6	rthomas	Rhian	Thomas	M	Dr	19/05/1988	04/03/2022	Employee	Llandegfan
7	Iglyn	Llion	Glyn	F	Mr	07/04/1982	19/11/2013	Employee	Llanbadrig
8	iglyn	Iwan	Glyn	F	Ms	15/12/1968	07/10/2012	Employee	Valley
9	aedwards	Aled	Edwards	F	Dr	18/06/1969	22/12/2019	Manager	Newborough
10	jjones	Jade	Jones	M	Dr	13/11/1983	01/05/2012	Employee	Bodedern
11	wmorgan	Wyn	Morgan	F	Dr	17/01/1961	23/08/2013	Manager	Llanbadrig
12	knia	Katie	Nia	M	Dr	08/09/1976	06/09/2022	Manager	Beaumaris
13	lrees	Llion	Rees	M	Ms	11/03/1977	12/10/2011	Employee	Llanfairpwllgwyngyll
14	ledwards	Llion	Edwards	M	Ms	09/08/1985	03/10/2013	Employee	Valley
15	m davies	Megan	Davies	F	Ms	02/11/1964	19/05/2018	Employee	Holyhead
16	cowen	Carys	Owen	F	Dr	14/09/1980	06/02/2012	Employee	Holyhead

From here the manager can choose to interact with the data from the “Employees” tab or change to another tab to interact with different data.

If choosing to interact with the data the manager can select a record to make changes to it:

ID	Username	Forename	Surname	Sex	Title	Birthdate	Hire Date	Job Description	Town
1	admin	None	None	None	None	None	None	Manager	None
2	employee	None	None	None	None	None	None	Employee	None
3	manager	Nia	Fychan	M	Dr	07/12/2000	02/06/2012	Manager	Newborough
4	alloyd	Aled	Lloyd	F	Ms	12/12/1961	27/02/2019	Employee	Rhosneigr
5	ndavies	Nia	Davies	F	Dr	18/07/1981	13/11/2012	Manager	Llanddona
6	rthomas	Rhian	Thomas	M	Dr	19/05/1988	04/03/2022	Employee	Llandegfan
7	lglyn	Llion	Glyn	F	Mr	07/04/1982	19/11/2013	Employee	Llanbadrig
8	iglyn	Iwan	Glyn	F	Ms	15/12/1968	07/10/2012	Employee	Valley
9	aedwards	Aled	Edwards	F	Dr	18/06/1969	22/12/2019	Manager	Newborough
10	ijones	Jade	Jones	M	Dr	13/11/1983	01/05/2012	Employee	Bodedern
11	wmorgan	Wyn	Morgan	F	Dr	17/01/1961	23/08/2013	Manager	Llanbadrig
12	knia	Katie	Nia	M	Dr	08/09/1976	06/09/2022	Manager	Beumaris
13	lrees	Llion	Rees	M	Ms	11/03/1977	12/10/2011	Employee	Llanfairpwllgwyngyll
14	ledwards	Llion	Edwards	M	Ms	09/08/1985	03/10/2013	Employee	Valley
15	mdavies	Megan	Davies	F	Ms	02/11/1964	19/05/2018	Employee	Holyhead
16	cowen	Carys	Owen	F	Dr	14/09/1980	06/02/2012	Employee	Holyhead

**Record**

ID:	4	Username:	alloyd	Password:	*****
Forename:	Aled	Surname:	Lloyd	Sex:	F
Title:	Ms	Birthdate:	12/12/1961	Hire Date:	27/02/2019
Job Description:	Employee	Town:	Rhosneigr	County:	Conwy
Postcode:	LL77 1ES	Phone Number:	09217399042	Email:	aled.lloyd@example.cc

**Range Search**

Birthdate		
After Date:	01/01/1920	Filter
Before Date:	01/01/2020	
Hire Date		
After Date:	01/01/1920	Filter
Before Date:	01/01/2020	

**Commands**

From here the commands at the bottom of the window can be used to commit changes or perform searches.

The “Range Search” frame on the right can also be used to perform searches on the data:

ID	Username	Forename	Surname	Sex	Title	Birthdate	Hire Date	Job Description	Town
15	mrdavies	Megan	Davies	F	Ms	02/11/1964	19/05/2018	Employee	Holyhead
185	nwalters	Nia	Walters	M	Dr	07/12/1987	05/11/2019	Manager	Holyhead

This example shows the Hire Date filter being applied between 2018 and 2020, as well as the filter of employees who are situated in Holyhead.

The different tabs across the top associate with different personal records that can be searched and edited:

Lizzy Lee's - Manager Dashboard

**Records**

	Employees	Teacher	Student	Customer					
1	teacher	Iurien	Price	F	Mr	11/03/1994	13/06/2010	Amlwch	Gwynedd
2	iprice	Iwan	Price	M	Mr	17/06/1969	08/01/2019	Llanbadrig	Anglesey
3	ania	Aled	Nia	M	Mr	21/02/1978	08/09/2016	Holyhead	Anglesey
4	dnia	Dafydd	Nia	M	Mr	09/12/2000	05/12/2010	Llanerchymedd	Conwy
5	iyorath	Iwan	Yorath	F	Dr	04/03/1992	15/06/2012	Llanerchymedd	Anglesey
6	gidris	Gareth	Idris	F	Dr	26/01/1968	03/02/2022	Gaerwen	Anglesey
7	mjones	Megan	Jones	M	Mr	14/01/1971	10/11/2018	Amlwch	Gwynedd
8	sceri	Sian	Ceri	F	Mr	16/01/1974	22/06/2022	Amlwch	Anglesey
9	ifychan	Iwan	Fychan	M	Mrs	18/08/1990	05/02/2019	Llandegfan	Gwynedd
10	kwalters	Katie	Walters	M	Ms	11/11/1982	14/09/2019	Llanerchymedd	Anglesey
11	uhedd	Urien	Hedd	M	Mr	01/04/1997	03/11/2014	Gaerwen	Anglesey
12	fthomas	Ffion	Thomas	F	Mr	14/03/1994	15/07/2016	Holyhead	Conwy
13	kroberts	Katie	Roberts	M	Mr	18/05/1981	02/12/2011	Beaumaris	Anglesey
14	aedwards	Aled	Edwards	M	Dr	22/10/1986	04/01/2022	Amlwch	Anglesey
15	uthomas	Urien	Thomas	M	Dr	02/03/1973	25/08/2017	Menai Bridge	Conwy
16	omorgan	Owain	Morgan	M	Dr	13/02/1963	10/09/2010	Rhosneigr	Gwynedd

**Stock**

**Sales**

**Commands**

[Clear Fields](#) [Update](#) [Search](#) [Add Record](#) [Delete Record](#)

Lizzy Lee's - Manager Dashboard

**Records**

	Employees	Teacher	Student	Customer						
1	student	Gethin	Edwards	F	Mr	13/03/1965	Pentraeth	Conwy	LL77 3FL	0
2	Wyn891	Wyn	Ceredig	M	Mr	21/01/1973	Menai Bridge	Gwynedd	LL77 6RP	0
3	Gethin345	Dyfan	Fychan	F	Ms	01/08/1971	Menai Bridge	Gwynedd	LL77 8CW	0
4	Urien196	Ffion	Fychan	F	Mrs	04/01/1991	Llangefni	Conwy	LL77 2AL	0
5	Haf461	Mabon	Jenkins	F	Mr	07/11/1961	Gaerwen	Anglesey	LL77 3JR	0
6	Cadell625	Cadell	Lloyd	F	Mrs	06/02/1966	Gaerwen	Anglesey	LL77 9SY	0
7	Wyn356	Ffion	Edwards	M	Mrs	12/08/1998	Rhosneigr	Anglesey	LL77 7EE	0
8	Ynyr340	Llio	Stephens	F	Mrs	25/07/1992	Amlwch	Anglesey	LL77 6UC	0
9	Gethin414	Ffion	Thomas	F	Ms	25/10/1991	Beaumaris	Anglesey	LL77 8NL	0
10	Nia799	Mabon	Jenkins	M	Mrs	10/08/1995	Llangefni	Gwynedd	LL77 6BA	0
11	Zara736	Nia	Fychan	M	Ms	25/02/1967	Amlwch	Gwynedd	LL77 9PU	0
12	Owain469	Haf	Edwards	M	Ms	06/11/1992	Holyhead	Gwynedd	LL77 9PA	0
13	Llio557	Dyfan	Edwards	F	Mrs	21/10/1994	Benllech	Gwynedd	LL77 3UW	0
14	Aneirin865	Branwen	Morgan	F	Mrs	25/03/1993	Beaumaris	Conwy	LL77 8YF	0
15	Branwen342	Branwen	Roberts	M	Mr	26/09/1981	Pentraeth	Conwy	LL77 3CN	0
16	Zara914	Dyfan	Morgan	F	Ms	03/12/1999	Pentraeth	Conwy	LL77 6YA	0

**Stock**

**Sales**

**Commands**

[Clear Fields](#) [Update](#) [Search](#) [Add Record](#) [Delete Record](#)

The screenshot shows a software application window titled "Lizzy Lee's - Manager Dashboard". The main area displays a grid of employee records with columns: ID, Forename, Surname, Sex, Title, Birthdate, Town, County, Postcode, Phone Number, and Email. The records list various employees such as Bryn, Ceri, Urien, Edwards, Huw, Yorath, Carys, Lloyd, Llion, Jones, Katie, Glyn, Dafydd, Morgan, Megan, Walters, Katie, Morgan, Bryn, Jones, Megan, Williams, Iwan, Hedd, Tegan, Edwards, and Bryn, Yorath. The interface includes a navigation bar with tabs for File, Edit, Employees, Teacher, Student, Customer, and a sidebar with links for Records, Lessons, Stock, and Sales. Below the grid are two search/filter panels: "Record" and "Range Search". The "Record" panel contains fields for ID, Forename, Surname, Sex, Title, Birthdate, Town, County, Postcode, Phone Number, and Email. The "Range Search" panel allows filtering by Birthdate, with fields for After Date (01/01/1920) and Before Date (01/01/2020), and a "Filter" button.

ID	Forename	Surname	Sex	Title	Birthdate	Town	County	Postcode	Phone Number	Email
1	Bryn	Ceri	F	Dr	19/03/1992	Y Felinheli	Conwy	LL77 4IW	09995109975	brynceri@lizzy.com
2	Bryn	Ceri	F	Ms	10/08/1991	Valley	Anglesey	LL77 3DC	07590141544	brynceri@lizzy.com
3	Urien	Edwards	M	Mrs	03/08/1971	Benllech	Conwy	LL77 4BL	09225810512	uriens.edwards@lizzy.com
4	Huw	Yorath	F	Dr	12/04/1996	Valley	Conwy	LL77 8MH	09154010070	huwyorath@lizzy.com
5	Carys	Yorath	F	Mr	06/01/2000	Moelfre	Conwy	LL77 4AN	08362122575	carysyorath@lizzy.com
6	Llion	Lloyd	M	Dr	06/04/1981	Bodedern	Anglesey	LL77 3TK	07277954378	llion.lloyd@lizzy.com
7	Llion	Jones	F	Mrs	06/12/1960	Valley	Anglesey	LL77 1HS	07098878777	llion.jones@lizzy.com
8	Katie	Glyn	M	Ms	13/11/2000	Llanerchymedd	Conwy	LL77 8HR	08049862526	katie.glyn@lizzy.com
9	Dafydd	Morgan	M	Mrs	05/11/1962	Benllech	Conwy	LL77 5BV	09608846219	dafydd.morgan@lizzy.com
10	Megan	Walters	M	Dr	15/01/1961	Gaerwen	Anglesey	LL77 4MY	08749139575	megan.walters@lizzy.com
11	Katie	Morgan	M	Dr	02/03/1975	Menai Bridge	Anglesey	LL77 1RY	08994247557	katie.morgan@lizzy.com
12	Bryn	Jones	F	Mr	27/12/1989	Llandegfan	Gwynedd	LL77 7OJ	08772684637	bryn.jones@lizzy.com
13	Megan	Williams	F	Mrs	23/05/1998	Trearddur Bay	Conwy	LL77 7PL	08992264628	megan.williams@lizzy.com
14	Iwan	Hedd	F	Mrs	23/03/1990	Menai Bridge	Anglesey	LL77 1SL	09057511314	iwan.hedd@lizzy.com
15	Tegan	Edwards	M	Mr	01/03/1993	Llanerchymedd	Conwy	LL77 9BC	07111670280	tegan.edwards@lizzy.com
16	Bryn	Yorath	F	Ms	03/01/1965	Y Felinheli	Conwy	LL77 9IQ	07884449396	bryn.yorath@lizzy.com

Down the left side of the screen the user can change between different sections of the system if their access rights allow it.

The “Lessons” section displays all records related to students lessons:

Lizzy Lee's - Manager Dashboard

**Records**

ID	Student	Teacher	Location	Lesson Frequency	Lesson Day
1	Llio Stephens	Dafydd Nia	Room 8	Monthly	Tuesday
2	Dyfan Ceredig	Katie Roberts	Room 7	Fortnightly	Tuesday
3	Dyfan Fychan	Owain Bevan	Room 3	Fortnightly	Friday
4	Owain Howells	Gareth Idris	Room 7	Monthly	Tuesday
5	Haf Lloyd	Iwan Yorath	Room 9	Weekly	Wednesday
6	Llio Khan	Iwan Fychan	Room 1	Fortnightly	Friday
7	Cadell Owen	Katie Walters	Room 3	Monthly	Thursday
8	Rhian Davies	Dafydd Nia	Room 2	Fortnightly	Thursday
9	Dyfan Williams	Aled Edwards	Room 5	Weekly	Monday
10	Ynwr Bevan	Iwan Yorath	Room 6	Fortnightly	Tuesday
11	Dyfan Owen	Iwan Price	Room 2	Weekly	Tuesday
12	Dyfan Ap Dafydd	Gareth Idris	Room 10	Weekly	Thursday
13	Mabon Khan	Iwan Yorath	Room 4	Fortnightly	Tuesday
14	Nia Thomas	Iwan Price	Room 10	Monthly	Monday
15	Mabon Khan	Iwan Fychan	Room 6	Fortnightly	Thursday
16	Dyfan Khan	Megan Jones	Room 6	Fortnightly	Wednesday

**Lessons**

**Stock**

**Sales**

**Record**

ID:  Student ID:  Teacher ID:   
 Location:  Lesson Frequency:  Lesson Day:   
 Lesson Time:  Lesson Length:  Lesson Cost:   
 Booking Date:  Last Update:  Cancelled:

**Export**

Calculate Clear  
Export as .txt file: Export

**Commands**

Clear Fields Update Search Add Record Delete Record

Lizzy Lee's - Manager Dashboard

**Records**

ID	Lesson Title	Student	Teacher	Attended	Student Behaviour
1	Singing Techniques a	Dyfan Edwards	Aled Nia	1	2
2	Advanced Drums and	Ffion Edwards	Aled Nia	0	2
3	Singing Techniques a	Sioned Griffiths	Tegan Fychan	0	0
4	Guitar Chords and Te	Mabon Jenkins	Iwan Yorath	0	3
5	Songwriting and Con	Wyn Khan	Urien Thomas	1	2
6	Guitar Chords and Te	Aneirin Williams	Iwan Price	1	3
7	Singing Techniques a	Wyn Stephens	Tegan Fychan	0	0
8	Music Theory for Beg	Urien Griffiths	Owain Bevan	1	0
9	Songwriting and Con	Urien Edwards	Tegan Fychan	0	4
10	Music Theory for Beg	Haf Lloyd	Dafydd Nia	1	4
11	Songwriting and Con	Cadell Morgan	Gareth Idris	1	2
12	Singing Techniques a	Mabon Price	Owain Morgan	0	4
13	Singing Techniques a	Llio Ap Dafydd	Iwan Fychan	0	0
14	Singing Techniques a	Llio Iorwerth	Megan Jones	1	4
15	Guitar Chords and Te	Rhian Davies	Katie Walters	1	4
16	Guitar Chords and Te	Mabon Bevan	Iwan Price	1	1

**Lessons**

**Stock**

**Sales**

**Record**

ID:  Lesson Plans ID:  Student ID:   
 Teacher ID:  Attended:  Student Behaviour:   
 Notes:  Date:

**Report Notes**

Note Commands  
Make changes to note: Edit Save Changes

**Commands**

Clear Fields Update Search Add Record Delete Record

Lizzy Lee's - Manager Dashboard

File Edit

Student Bookings Lesson Report Lesson Plan

ID	Lesson Title	Lesson Objective
1	Guitar Chords and Techniques	Understanding music notation and basic theory co
2	Songwriting and Composition	Mastering complex drum patterns and rhythm
3	Singing Techniques and Performance	Learning basic chords and strumming patterns on
4	Singing Techniques and Performance	Learning basic chords and strumming patterns on
5	Music Theory for Beginners	Mastering complex drum patterns and rhythm
6	Guitar Chords and Techniques	Mastering complex drum patterns and rhythm
7	Guitar Chords and Techniques	Learning basic chords and strumming patterns on
8	Songwriting and Composition	Exploring different songwriting techniques and app
9	Singing Techniques and Performance	Exploring different songwriting techniques and app
10	Singing Techniques and Performance	Learning basic chords and strumming patterns on
11	Music Theory for Beginners	Exploring different songwriting techniques and app
12	Singing Techniques and Performance	Exploring different songwriting techniques and app
13	Guitar Chords and Techniques	Exploring different songwriting techniques and app
14	Music Theory for Beginners	Improving vocal range and control for better singin
15	Songwriting and Composition	Mastering complex drum patterns and rhythm
16	Guitar Chords and Techniques	Understanding music notation and basic theory co

Records Lessons Stock Sales

Lesson procedure

Procedure Commands  
Make changes to procedure: [Edit](#) [Save Changes](#)

Record

ID:	<input type="text"/>	Lesson Title:	<input type="text"/>	Lesson Objective:	<input type="text"/>
Materials:	<input type="text"/>	Procedure:	<input type="text"/>		

Commands

[Clear Fields](#) [Update](#) [Search](#) [Add Record](#) [Delete Record](#)

The “Stock” section displays all the records related to stock:

Lizzy Lee's - Manager Dashboard

**Records**

ID	Brand	Type	Model	Price	Serial Num
1	Epiphone	Electric	Les Paul	1369.49	8V3V8PJA9K
2	Schecter	Classical	JEM	748.21	7VDP6MMST7P
3	Jackson	Bass	Warlock	555.35	WRHAUNQAHM
4	Ibanez	Acoustic	JEM	541.52	CJ9SSEIKMC
5	Schecter	Acoustic	PRS Custom 24	733.11	XEN3D2YVVI
6	PRS	Electric	Flying V	1837.36	813NPYSLRF
7	Epiphone	Classical	Telecaster	1430.52	Y7527F4SU3
8	Jackson	Bass	Warlock	174.1	T66CHZSL3
9	Fender	Classical	Les Paul	764.23	UO7ZXJ6CHX
10	Epiphone	Electric	JEM	1685.12	CSTKGMIION
11	ESP	Bass	SG	1340.37	HS0BXIC4OL
12	Schecter	Acoustic	Stratocaster	79.27	KJUTR84LNX
13	Schecter	Acoustic	Telecaster	719.72	QQYBFEP00A
14	ESP	Electric	Warlock	485.77	CX51UCOU2L
15	Epiphone	Acoustic	JEM	1919.88	I8A41TJURV
16	PRS	Electric	PRS Custom 24	1561.16	42AIECBILV

**Stock**

Record

ID:	Brand:	Type:
Model:	Price:	Serial Num:

Range Search

Price

Above:	Below:	Filter
--------	--------	--------

**Sales**

Commands

**Clear Fields** **Update** **Search** **Add Record** **Delete Record**

Lizzy Lee's - Manager Dashboard

**Records**

ID	Type	Description	Price	Expiry Date
1	Cookie	Thumbprint Cookie	2.25	12/09/2029
2	Pastry	Pain au Raisin	1.2	17/06/2027
3	Salad	Seafood Salad	11.31	15/09/2020
4	Soda	Cactus Cooler	7.3	09/12/2020
5	Salad	Somen Salad	12.96	28/08/2021
6	Tea	Teh Chai	19.5	26/11/2027
7	Pastry	Pain au Lait	7.22	19/05/2024
8	Pastry	Pain au Levain	3.98	15/12/2026
9	Cookie	Peanut Butter Blossor	18.91	20/02/2027
10	Pastry	Pain aux Raisins	2.06	17/08/2023
11	Hot Drink	Chocolate Milk	6.46	14/06/2024
12	Cookie	Fortune Cookie	3.79	20/04/2024
13	Soda	Cherry 7 Up	3.25	27/07/2031
14	Salad	Carrot Salad	19.65	18/05/2022
15	Soda	Cherry A&W Root Be	17.22	08/02/2025
16	Soda	Diet Coke Raspberry	1.37	05/04/2027
17	Salad	Tabbouleh	7.04	12/10/2032

**Stock**

Record

ID:	Type:	Description:
Price:	Expiry Date:	

Range Search

Price

Above:	Below:	Filter
--------	--------	--------

**Sales**

Commands

**Clear Fields** **Update** **Search** **Add Record** **Delete Record**

Lizzy Lee's - Manager Dashboard

File Edit

Music Item Cafe Item Music Inventory Cafe Inventory

ID	Music Item	Count	Date
1	Warlock	9	14/05/2021
2	SG	10	16/08/2022
3	Explorer	7	11/03/2019
4	SG	7	01/01/2022
5	PRS Custom 24	2	08/12/2020
6	JEM	13	10/02/2022
7	Telecaster	12	23/03/2022
8	PRS Custom 24	3	19/05/2021
9	Les Paul	6	21/11/2020
10	PRS Custom 24	15	05/10/2018
11	SG	6	17/02/2021
12	Les Paul	13	09/11/2022
13	Stratocaster	9	16/10/2021
14	Explorer	5	26/04/2020
15	PRS Custom 24	2	22/10/2018
16	JEM	2	24/11/2018

Records Lessons Stock Sales

Inventory Estimate

Record

ID:  Music Item ID:  Count:   
Date:

Commands

Clear Fields Update Search Add Record Delete Record

Lizzy Lee's - Manager Dashboard

File Edit

Music Item Cafe Item Music Inventory Cafe Inventory

ID	Cafe Item	Count	Date
1	Pain au Lard	7	09/12/2021
2	Chocolate Chai	10	09/06/2018
3	Teh Halia	2	09/03/2019
4	Diet Coke	4	08/09/2020
5	A&W Root Beer	3	07/12/2019
6	Pain	5	13/03/2020
7	Diet Coke Lemon	8	11/05/2018
8	Biscotti	11	18/04/2022
9	Cafe Con Leche	8	03/03/2021
10	Pain au Lard	7	26/01/2020
11	Caffe Mocha	2	08/04/2020
12	German Chocolate C.	11	24/03/2021
13	Biscotti	9	07/01/2018
14	Diet Coke Cherry	15	14/06/2022
15	Diet Coke Raspberry	11	23/06/2020
16	Pain au Lard	8	06/09/2020

Records Lessons Stock Sales

Inventory Estimate

Record

ID:  Cafe Item ID:  Count:   
Date:

Commands

Clear Fields Update Search Add Record Delete Record

And finally the “Sales” section displays all records of sales:

Lizzy Lee's - Manager Dashboard

**Records**

**Lessons**

**Stock**

**Sales**

**Music Sales** **Cafe Sales**

ID	Music Item	Customer	Count	Date
1	Flying V	Katie Bevan	3	20/10/2021
2	JEM	Owain Idris	1	04/05/2019
3	JEM	Wyn Roberts	1	11/06/2019
4	SG	Sian Walters	2	01/12/2022
5	Explorer	Elin Roberts	1	22/12/2020
6	Stratocaster	Elin Lloyd	3	26/05/2019
7	Warlock	Huw Morgan	1	24/05/2018
8	Les Paul	Sian Yorath	3	03/10/2019
9	Stratocaster	Owain Price	1	07/02/2020
10	JEM	Ffion Davies	1	19/10/2021
11	Explorer	Huw Morgan	1	26/02/2018
12	JEM	Iwan Ceri	2	21/04/2020
13	Les Paul	Huw Fychan	1	18/08/2019
14	Explorer	Wyn Davies	1	15/10/2020
15	Telecaster	Owain Price	1	16/01/2022
16	PRS Custom 24	Jade Rees	3	03/03/2018

**Sales Estimate**

**Record**

ID:  Music Item ID:  Customer ID:   
 Count:  Date:

**Commands**

**Clear Fields** **Update** **Search** **Add Record** **Delete Record**

Lizzy Lee's - Manager Dashboard

**Records**

**Lessons**

**Stock**

**Sales**

**Music Sales** **Cafe Sales**

ID	Cafe Item	Customer	Count	Date
1	Irish Coffee	Urien Idris	7	19/06/2020
2	Teh Halia	Ffion Owen	2	25/03/2022
3	Viennese Coffee	Dafydd Idris	3	24/12/2019
4	Pain au Fromage	Iwan Owen	7	02/10/2020
5	Chocolate Coffee	Tegan Walters	4	24/12/2020
6	Cafe Noisette	Iwan Thomas	10	10/07/2020
7	Tabbouleh Salad	Owain Price	7	15/04/2022
8	Oatmeal Cookie	Carys Edwards	3	11/05/2020
9	Pain au Levain	Megan Rees	2	12/07/2022
10	Teh Putih	Dafydd Davies	4	26/01/2022
11	Lobster Bisque	Nia Lloyd	3	15/05/2022
12	Teh Thailand	Bryn Jones	4	20/03/2020
13	Wedding Cake	Katie Jones	4	23/10/2021
14	Cafe Breva	Carys Walters	10	19/06/2019
15	Shopska Salad	Owain Lloyd	5	06/07/2022
16	Italian Wedding Soup	Aled Bevan	5	24/12/2022

**Sales Estimate**

**Record**

ID:  Cafe Item ID:  Customer ID:   
 Count:  Date:

**Commands**

**Clear Fields** **Update** **Search** **Add Record** **Delete Record**

The “Student Bookings” tab in “Lessons” has a unique frame that displays an estimate of the revenue calculated from the selected lessons. The estimate can also be exported to a .txt file if the user wants to store the information:

ID	Student	Teacher	Location	Lesson Frequency	Lesson Day	Lesson Time	Lesson Length
1	Llio Stephens	Dafydd Nia	Room 8	Monthly	Tuesday	08:41	60
2	Dyfan Ceredig	Katie Roberts	Room 7	Fortnightly	Tuesday	12:15	30
3	Dyfan Fychan	Owain Bevan	Room 3	Fortnightly	Friday	11:40	45
4	Owain Howells	Gareth Idris	Room 7	Monthly	Tuesday	08:39	60
5	Haf Lloyd	Iwan Yorath	Room 9	Weekly	Wednesday	13:01	60
6	Llio Khan	Iwan Fychan	Room 1	Fortnightly	Friday	08:25	60
7	Cadell Owen	Katie Walters	Room 3	Monthly	Thursday	09:17	30
8	Rhian Davies	Dafydd Nia	Room 2	Fortnightly	Thursday	12:15	30
9	Dyfan Williams	Aled Edwards	Room 5	Weekly	Monday	14:32	45
10	Ynry Bevan	Iwan Yorath	Room 6	Fortnightly	Tuesday	11:13	60
11	Dyfan Owen	Iwan Price	Room 2	Weekly	Tuesday	12:22	60
12	Dyfan Ap Dafydd	Gareth Idris	Room 10	Weekly	Thursday	09:20	30
13	Mabon Khan	Iwan Yorath	Room 4	Fortnightly	Tuesday	16:12	45
14	Nia Thomas	Iwan Price	Room 10	Monthly	Monday	14:22	60
15	Mabon Khan	Iwan Fychan	Room 6	Fortnightly	Thursday	12:50	30
16	Dyfan Khan	Megan Jones	Room 6	Fortnightly	Wednesday	15:02	45

User ID '2' has Fortnightly monthly lessons, each costing £35 resulting in a monthly revenue of: £70.0  
User ID '3' has Fortnightly monthly lessons, each costing £40 resulting in a monthly revenue of: £80.0  
User ID '4' has Monthly monthly lessons, each costing £20 resulting in a monthly revenue of: £40.0  
User ID '5' has Weekly monthly lessons, each costing £30 resulting in a monthly revenue of: £15.0  
User ID '6' has Fortnightly monthly lessons, each costing £35 resulting in a monthly revenue of: £35.0

Total Monthly Estimate for Selected Lessons: £345.0

Export:

Commands:

The output file looks like this (multiple calculations have been exported already)

```

LessonEstimate.txt
1 =====
2 06/05/2023: STUDENT_BOOKING Monthly Revenue Estimate:
3 User ID '2' has Monthly monthly lessons, each costing £15 resulting in a monthly revenue of: £15.0
4 User ID '3' has Weekly monthly lessons, each costing £20 resulting in a monthly revenue of: £80.0
5 User ID '4' has Fortnightly monthly lessons, each costing £20 resulting in a monthly revenue of: £40.0
6 User ID '5' has Fortnightly monthly lessons, each costing £10 resulting in a monthly revenue of: £20.0
7 User ID '6' has Fortnightly monthly lessons, each costing £10 resulting in a monthly revenue of: £20.0
8
9 Total Monthly Estimate for Selected Lessons: £175.0
10 =====
11 07/05/2023: STUDENT_BOOKING Monthly Revenue Estimate:
12 User ID '4' has Fortnightly monthly lessons, each costing £20 resulting in a monthly revenue of: £40.0
13 User ID '5' has Fortnightly monthly lessons, each costing £10 resulting in a monthly revenue of: £20.0
14 User ID '6' has Fortnightly monthly lessons, each costing £10 resulting in a monthly revenue of: £20.0
15
16 Total Monthly Estimate for Selected Lessons: £80.0
17 =====
18 10/05/2023: STUDENT_BOOKING Monthly Revenue Estimate:
19 User ID '2' has Fortnightly monthly lessons, each costing £35 resulting in a monthly revenue of: £70.0
20 User ID '3' has Fortnightly monthly lessons, each costing £40 resulting in a monthly revenue of: £80.0
21 User ID '4' has Monthly monthly lessons, each costing £40 resulting in a monthly revenue of: £40.0
22 User ID '5' has Weekly monthly lessons, each costing £30 resulting in a monthly revenue of: £120.0
23 User ID '7' has Monthly monthly lessons, each costing £35 resulting in a monthly revenue of: £35.0
24
25 Total Monthly Estimate for Selected Lessons: £345.0
26

```

The “Lesson Report” and “Lesson Plan” tabs also have an additional frame that displays long text from the record selected:

ID	Lesson Title	Lesson Objective
1	Guitar Chords and Techniques	Understanding music notation and basic theory co
2	Songwriting and Composition	Mastering complex drum patterns and rhythm
3	Singing Techniques and Performance	Learning basic chords and strumming patterns on
4	Singing Techniques and Performance	Learning basic chords and strumming patterns on
5	Music Theory for Beginners	Mastering complex drum patterns and rhythm
6	Guitar Chords and Techniques	Mastering complex drum patterns and rhythm
7	Guitar Chords and Techniques	Learning basic chords and strumming patterns on
8	Songwriting and Composition	Exploring different songwriting techniques and app
9	Singing Techniques and Performance	Exploring different songwriting techniques and app
10	Singing Techniques and Performance	Learning basic chords and strumming patterns on
11	Music Theory for Beginners	Exploring different songwriting techniques and app
12	Singing Techniques and Performance	Exploring different songwriting techniques and app
13	Guitar Chords and Techniques	Exploring different songwriting techniques and app
14	Music Theory for Beginners	Improving vocal range and control for better singir
15	Songwriting and Composition	Mastering complex drum patterns and rhythm
16	Guitar Chords and Techniques	Understanding music notation and basic theory co

**Lesson procedure**  
Begin by reviewing basic finger and hand position on the keyboard, then practice playing simple melodies and scales. Move on to more complex pieces as proficiency increases.

**Procedure Commands**  
Make changes to procedure: [Edit](#) [Save Changes](#)

ID	Lesson Title	Student	Teacher	Attended	Student Behaviour
1	Singing Techniques a	Dyfan Edwards	Aled Nia	1	2
2	Advanced Drums anc	Ffion Edwards	Aled Nia	0	2
3	Singing Techniques a	Sioned Griffiths	Tegan Fychan	0	0
4	Guitar Chords and Te	Mabon Jenkins	Iwan Yorath	0	3
5	Songwriting and Con	Wyn Khan	Urien Thomas	1	2
6	Guitar Chords and Te	Aneirin Williams	Iwan Price	1	3
7	Singing Techniques a	Wyn Stephens	Tegan Fychan	0	0
8	Music Theory for Beg	Urien Griffiths	Owain Bevan	1	0
9	Songwriting and Con	Urien Edwards	Tegan Fychan	0	4
10	Music Theory for Beg	Haf Lloyd	Dafydd Nia	1	4
11	Songwriting and Con	Cadell Morgan	Gareth Idris	1	2
12	Singing Techniques a	Mabon Price	Owain Morgan	0	4
13	Singing Techniques a	Llio Ap Dafydd	Iwan Fychan	0	0
14	Singing Techniques a	Llio Iorwerth	Megan Jones	1	4
15	Guitar Chords and Te	Rhian Davies	Katie Walters	1	4
16	Guitar Chords and Te	Mabon Bevan	Iwan Price	1	1
17	Songwriting and Con	Llio Roberts	Iwan Fychan	0	0

**Report Notes**  
Dyfan has grown significantly today. His confidence has really begun to shine

**Note Commands**  
Make changes to note: [Edit](#) [Save Changes](#)

The text can be edited and saved like normal fields in the Record Frame can

The “Sales” and “Inventory” tabs in the “Sales” and “Stock” sections respectively also have an additional frame that features identically to each other.

When the user selects one or more records from the treeview table the frame will display an estimated value of the items selected. This can be used to gain an estimate for the value of stock in the cafe’s inventory, or to value the revenue gained from sales at certain dates or from certain items:

ID	Music Item	Count	Date
1	Warlock	9	14/05/2021
2	SG	10	16/08/2022
3	Explorer	7	11/03/2019
4	SG	7	01/01/2022
5	PRS Custom 24	2	08/12/2020
6	JEM	13	10/02/2022
7	Telecaster	12	23/03/2022
8	PRS Custom 24	3	19/05/2021
9	Les Paul	6	21/11/2020
10	PRS Custom 24	15	05/10/2018
11	SG	6	17/02/2021
12	Les Paul	13	09/11/2022
13	Stratocaster	9	16/10/2021
14	Explorer	5	26/04/2020
15	PRS Custom 24	2	22/10/2018
16	JEM	2	24/11/2018

Lizzy Lee's - Manager Dashboard

File Edit

Records Lessons Stock Sales

Music Item Cafe Item Music Inventory Cafe Inventory

ID	Cafe Item	Count	Date
456	Biscotti	9	02/01/2018
64	Cafe Con Miel	11	02/01/2018
74	Cafe Del Tiempo	15	03/01/2018
221	Biscotti	5	05/01/2018
13	Biscotti	9	07/01/2018
120	Irish Coffee	6	14/01/2018
368	Diet Coke Vanilla	12	17/01/2018
106	French Onion Soup	12	18/01/2018
405	Clam Chowder	8	20/01/2018
284	Pain au Levain	15	28/01/2018
25	Pepsi	11	28/01/2018
404	Pain au Lait	9	01/02/2018
304	Corn Chowder	6	02/02/2018
273	Pastrami Sandwich	3	06/02/2018
388	Molasses Cookie	6	07/02/2018
407	Cheese Sandwich	15	10/02/2018

Inventory Estimate  
Estimated inventory value: \$898.56

Record

ID:	456	Cafe Item ID:	26	Count:	9
Date:	02/01/2018				

Commands

Clear Fields Update Search Add Record Delete Record

Lizzy Lee's - Manager Dashboard

File Edit

Records Lessons Stock Sales

Music Sales Cafe Sales

ID	Cafe Item	Customer	Count	Date
771	Ambrosia Salad	Elin Williams	6	04/08/2018
745	Ambrosia Salad	Rhian Edwards	5	16/06/2020
614	Ambrosia Salad	Jade Hedd	4	15/11/2019
398	Ambrosia Salad	Megan Rees	9	18/12/2022
77	Ambrosia Salad	Gareth Ceri	3	20/01/2021
999	Americano	Aled Edwards	4	11/09/2019
991	Americano	Jade Morgan	5	21/08/2022
979	Americano	Nia Glyn	6	14/01/2018
436	Americano	Huw Yorath	7	08/12/2020
108	Americano	Rhian Rees	9	19/11/2022
807	Angel Food Cake	Wyn Morgan	6	21/09/2018
630	Angel Food Cake	Iwan Hedd	2	28/03/2018
560	Angel Food Cake	Iwan Lloyd	4	05/12/2018
523	Angel Food Cake	Ffion Davies	1	12/02/2019
978	Bacon Sandwich	Dafydd Yorath	3	22/04/2020
846	Bacon Sandwich	Rhian Fychan	4	09/04/2020

Sales Estimate  
Total Revenue: £211.73

Record

ID:	999	Cafe Item ID:	245	Customer ID:	358
Count:	4	Date:	11/09/2019		

Commands

Clear Fields Update Search Add Record Delete Record

Managers and Employees can also import .csv files into a table, and also export a table to a .csv file.

This is achieved by selecting the Import/Export button under Edit in the toolbar:

ID	Cafe Item	Customer	Count	Date
771	Ambrosia Salad	Elin Williams	6	04/08/2018
745	Ambrosia Salad	Rhian Edwards	5	16/06/2020
614	Ambrosia Salad	Jade Hedd	4	15/11/2019
398	Ambrosia Salad	Megan Rees	9	18/12/2022
77	Ambrosia Salad	Gareth Ceri	3	20/01/2021
999	Americano	Aled Edwards	4	11/09/2019
991	Americano	Jade Morgan	5	21/08/2022
979	Americano	Nia Glyn	6	14/01/2018
436	Americano	Huw Yorath	7	08/12/2020
108	Americano	Rhian Rees	9	19/11/2022
807	Angel Food Cake	Wyn Morgan	6	21/09/2018
630	Angel Food Cake	Iwan Hedd	2	28/03/2018
560	Angel Food Cake	Iwan Lloyd	4	05/12/2018
523	Angel Food Cake	Ffion Davies	1	12/02/2019
978	Bacon Sandwich	Dafydd Yorath	3	22/04/2020
846	Bacon Sandwich	Rhian Fychan	4	09/04/2020

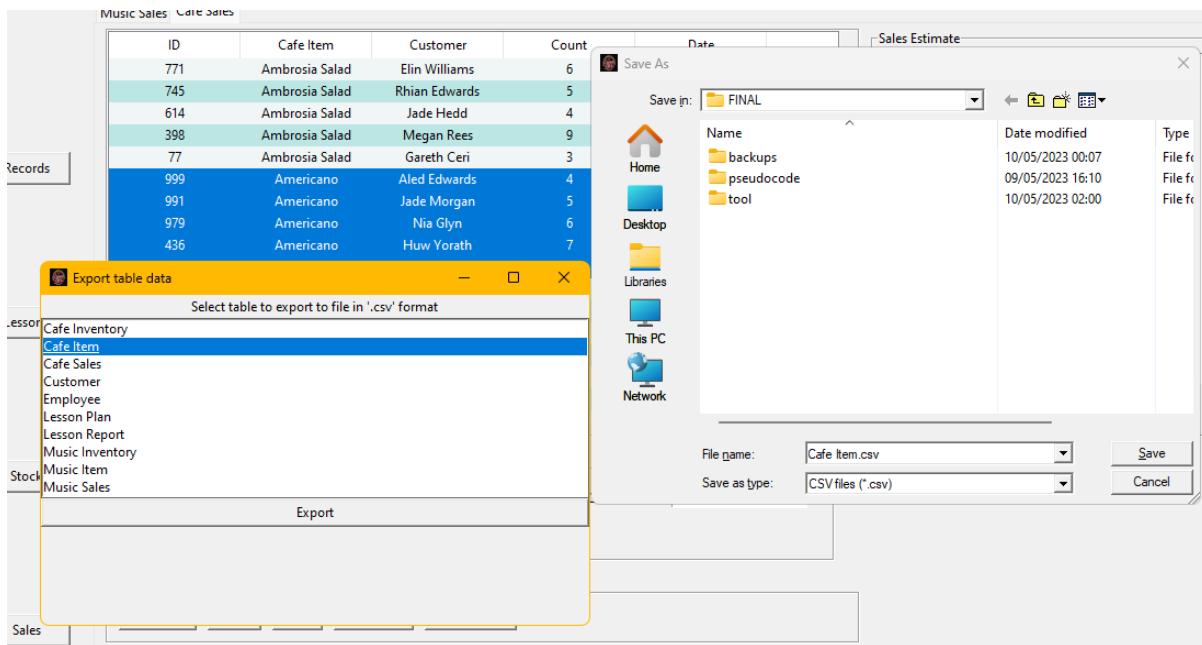
The Import window will allow the user to select a table and a file to perform the import from:

Import table data

Select '.csv' file and table to import data to

Cafe Inventory  
Cafe Item  
Cafe Sales  
Customer  
Employee  
Lesson Plan  
Lesson Report  
Music Inventory  
Music Item  
Music Sales

And the Export window works similarly by allowing the user to select a table and then choose the filename to export to:



.csv files will look like this:

The first line must be the column header for the data below it.

```
tool > Employee.csv
1 username,password,forename,surname,sex,title,birthdate,hire_date,job_description,town,county,postcode,phone_number,email
2 bceri,Passw0rd!30,Brynn,Ceri,F,Ms,03/08/1981,11/04/2011,Employee,Rhosneigr,Anglesey,LL77 5CQ,07481334691,brynn.ceri@example.com
3 nfychan,Passw0rd!44,Nia,Fychan,M,Dr,07/12/2000,02/06/2012,Employee,Newborough,Conwy,LL77 5MX,08197476819,nia.fychan@example.com
4 alloyd,Passw0rd!3,Aled,Lloyd,F,Ms,10/10/1987,27/02/2019,Employee,Rhosneigr,Conwy,LL77 1ES,09217399042,aled.lloyd@example.com
5 ndavies,Passw0rd!29,Nia,Davies,F,Dr,18/07/1981,13/11/2012,Manager,Llanddona,Gwynedd,LL77 7FA,07833665213,nia.davies@example.com
6 rthomas,Passw0rd!85,Rhian,Thomas,M,Dr,19/05/1988,04/03/2022,Employee,Llandegfan,Gwynedd,LL77 8TQ,07086894092,rhian.thomas@example.com
7 lglyn,Passw0rd!36,Llion,Glyn,F,Mr,07/04/1982,19/11/2013,Employee,Llanbadrig,Anglesey,LL77 6NT,08129578817,llion.glyn@example.com
8 iglyn,Passw0rd!77,Iwan,Glyn,F,Ms,15/12/1968,07/10/2012,Employee,Valley,Gwynedd,LL77 6WX,07738885422,iwan.glyn@example.com
9 aedwards,Passw0rd!62,Aled,Edwards,F,Dr,18/06/1969,22/12/2019,Manager,Newborough,Gwynedd,LL77 3ZP,09206055422,aled.edwards@example.com
10 jjones,Passw0rd!18,Jade,Jones,M,Dr,13/11/1983,01/05/2012,Employee,Bodedern,Anglesey,LL77 7AC,08682201901,jade.jones@example.com
11 wmorgan,Passw0rd!91,Wyn,Morgan,F,Dr,17/01/1961,23/08/2013,Manager,Llanbadrig,Gwynedd,LL77 3YF,07062856597,wyn.morgan@example.com
12 knia,Passw0rd!4,Katie,Nia,M,Dr,08/09/1976,06/09/2022,Manager,Beumaris,Gwynedd,LL77 3HQ,07925838102,katie.nia@example.com
13 lrees,M,Ms,11/03/1977,12/10/2011,Employee,Llanfairpwllgwyngyll,Conwy,LL77 5WL,07307690104,llion.rees@example.com
14 ledwards,Passw0rd!26,Llion,Edwards,M,Ms,09/08/1985,03/10/2013,Employee,Valley,Conwy,LL77 3HR,07968321196,llion.edwards@example.com
15 mdavies,Passw0rd!45,Megan,Davies,F,Ms,02/11/1964,19/05/2018,Employee,Holyhead,Conwy,LL77 6HV,08924076011,megan.davies@example.com
16 cowen,Passw0rd!40,Carys,Owen,F,Dr,14/09/1980,06/02/2012,Employee,Holyhead,Anglesey,LL77 1EY,07215959347,carys.owen@example.com
17 hwalters,Passw0rd!11,Huw,Walters,M,Dr,03/01/1970,03/05/2021,Manager,Menai Bridge,Gwynedd,LL77 9HA,09231510521,huw.walters@example.com
18 eroberts,Passw0rd!42,Elin,Roberts,F,Mrs,26/11/1988,03/06/2021,Manager,Llanerchymedd,Gwynedd,LL77 1ER,08402728038,elin.roberts@example.com
19 shedd,Passw0rd!67,Sian,Hedd,F,Mr,10/06/1982,13/08/2015,Manager,Llanddona,Conwy,LL77 3HK,09281210459,sian.hedd@example.com
20 mjones,Passw0rd!80,Megan,Jones,M,Mr,28/02/1976,09/01/2013,Manager,Valley,Gwynedd,LL77 9PD,09817492091,megan.jones@example.com
21 kroberts,Passw0rd!91,Katie,Roberts,F,Mrs,14/10/1988,22/04/2010,Manager,Cemaes,Gwynedd,LL77 9CE,09458330973,katie.roberts@example.com
22 fidris,Passw0rd!26,Ffion,Idris,F,Mrs,25/12/1988,15/01/2017,Employee,Pentraeth,Gwynedd,LL77 60V,07666323714,ffion.idris@example.com
23 hglyn,Passw0rd!72,Huw,Glyn,F,Mrs,16/02/1990,11/06/2017,Employee,Rhosneigr,Anglesey,LL77 7RA,09254281046,huw.glyn@example.com
24 iwalters,Passw0rd!27,Iwan,Walters,M,Dr,05/10/1979,04/08/2017,Employee,GAerwen,Conwy,LL77 6XS,08730629314,iwan.walters@example.com
25 sroberts,Passw0rd!19,Sian,Roberts,F,Ms,21/07/1965,02/12/2010,Manager,Llanerchymedd,Anglesey,LL77 8PC,08272494480,sian.roberts@example.com
26 rwalters,Passw0rd!90,Rhian,Walters,F,Mrs,20/01/1989,20/04/2018,Manager,Beumaris,Gwynedd,LL77 4BA,09372327650,rhian.walters@example.com
27 thedd,Passw0rd!90,Tegan,Hedd,M,Dr,04/07/1976,14/04/2011,Manager,Beumaris,Anglesey,LL77 7CE,08594911783,tegan.hedd@example.com
28 ayorath,Passw0rd!95,Aled,Yorath,M,Mr,13/06/1966,07/02/2011,Manager,Cemaes,Anglesey,LL77 5NZ,09046107776,aled.yorath@example.com
29 gyorath,Passw0rd!97,Gareth,Yorath,M,Dr,07/11/1993,17/01/2014,Employee,Llanfairpwllgwyngyll,Conwy,LL77 2FA,08060452736,gareth.yorath@example.com
30 rfychan,Passw0rd!98,Rhian,Fychan,F,Ms,06/03/1967,03/01/2015,Manager,Holyhead,Gwynedd,LL77 7IU,07071458506,rhian.fychan@example.com
31 rwalters,Passw0rd!52,Rhian,Walters,M,Mrs,07/12/1998,05/12/2017,Employee,Llanbadrig,Gwynedd,LL77 4FW,09216345152,rhian.walters@example.com
32 mprice,Passw0rd!41,Megan,Price,F,Mrs,13/07/1992,08/10/2020,Employee,Menai Bridge,Gwynedd,LL77 8VL,09653949326,megan.price@example.com
33 cowen,Passw0rd!57,Carys,Owen,M,Mr,05/01/1995,24/02/2012,Manager,Menai Bridge,Gwynedd,LL77 3TN,07606844644,carys.owen@example.com
```

Finally, the system supports providing users with separate access levels.

All the windows shown so far have been demonstrated with “Manager” access.

“Employees” will have the following accessible to them:

The screenshot shows a software application window titled "Lizzy Lee's - Employee Dashboard". The main area displays a grid of employee records with the following data:

ID	Username	Forename	Surname	Sex	Title	Birthdate	Hire Date	Job Description	Town
2	employee	None	None	None	None	None	None	Employee	None

Below the grid, there are several sections:

- Stock:** Contains fields for ID, Username, Password, Forename, Surname, Sex, Title, Birthdate, Hire Date, Job Description, Town, County, Postcode, Phone Number, and Email.
- Sales:** Contains fields for Birthdate, After Date (01/01/1920), Before Date (01/01/2020), and a Filter button.
- Range Search:** Contains fields for Birthdate, After Date (01/01/1920), Before Date (01/01/2020), and a Filter button.
- Commands:** Contains buttons for Clear Fields and Update.

Only their own records are visible and editable by them in the “Employee” tab

Lizzy Lee's - Employee Dashboard

File Edit

Employees Customer

ID	Forename	Surname	Sex	Title	Birthdate	Town	County	Postcode	Phone Number
1	Bryn	Ceri	F	Dr	19/03/1992	Y Felinheli	Conwy	LL77 4IW	09995109975 brync.
2	Bryn	Ceri	F	Ms	10/08/1991	Valley	Anglesey	LL77 3DC	07590141544 brynd.
3	Urien	Edwards	M	Mrs	03/08/1971	Benllech	Conwy	LL77 4BL	09225810512 urien.
4	Huw	Yorath	F	Dr	12/04/1996	Valley	Conwy	LL77 8MH	09154010070 huw.y.
5	Carys	Yorath	F	Mr	06/01/2000	Moelfre	Conwy	LL77 4AN	08362122575 carys.
6	Llion	Lloyd	M	Dr	06/04/1981	Bodedern	Anglesey	LL77 3TK	07277954378 llion.l
7	Llion	Jones	F	Mrs	06/12/1960	Valley	Anglesey	LL77 1HS	07098878777 llion.j
8	Katie	Glyn	M	Ms	13/11/2000	Llanerchymedd	Conwy	LL77 8HR	08049862526 katie.k.
9	Dafydd	Morgan	M	Mrs	05/11/1962	Benllech	Conwy	LL77 5BV	09608846219 dafydd.
10	Megan	Walters	M	Dr	15/01/1961	Gaerwen	Anglesey	LL77 4MY	08749139575 mega.w.
11	Katie	Morgan	M	Dr	02/03/1975	Menai Bridge	Anglesey	LL77 1RY	08994247557 katie.m.
12	Bryn	Jones	F	Mr	27/12/1989	Llandegfan	Gwynedd	LL77 7OJ	08772684637 brynj.
13	Megan	Williams	F	Mrs	23/05/1998	Trearddur Bay	Conwy	LL77 7PL	08992264628 mega.w.
14	Iwan	Hedd	F	Mrs	23/03/1990	Menai Bridge	Anglesey	LL77 1SL	09057511314 iwan.i.
15	Tegan	Edwards	M	Mr	01/03/1993	Llanerchymedd	Conwy	LL77 9BC	07111670280 tegan.e.
16	Bryn	Yorath	F	Ms	03/01/1965	Y Felinheli	Conwy	LL77 9IQ	07884449396 bryny.

Records Stock Sales

Record Range Search

Birthdate

After Date: 01/01/1920 Filter Before Date: 01/01/2020

Commands

Clear Fields Update Search Add Record Delete Record

They have full access to customer data.

Lizzy Lee's - Employee Dashboard

File Edit

Music Item Cafe Item Music Inventory Cafe Inventory

ID	Brand	Type	Model	Price	Serial Num
1	Epiphone	Electric	Les Paul	1369.49	8V3V8PJA9K
2	Schecter	Classical	JEM	748.21	7VDP6MMST7P
3	Jackson	Bass	Warlock	555.35	WRHAUNQAHM
4	Ibanez	Acoustic	JEM	541.52	CJ9SSIEKMC
5	Schecter	Acoustic	PRS Custom 24	733.11	XEN3D2YVVI
6	PRS	Electric	Flying V	1837.36	813NPYSLRF
7	Epiphone	Classical	Telecaster	1430.52	Y7527F4SU3
8	Jackson	Bass	Warlock	174.1	T66C1Z2SL3
9	Fender	Classical	Les Paul	764.23	UO7ZXJ6CHX
10	Epiphone	Electric	JEM	1685.12	CSTKGMIION
11	ESP	Bass	SG	1340.37	HSOBXIC4OL
12	Schecter	Acoustic	Stratocaster	79.27	KJUTR84LNX
13	Schecter	Acoustic	Telecaster	719.72	QQYBFEP00A
14	ESP	Electric	Warlock	485.77	CX51UC0U2L
15	Epiphone	Acoustic	JEM	1919.88	I8A41TJURV
16	PRS	Electric	PRS Custom 24	1561.16	42AIECBILV

Records Stock Sales

Record Range Search

Price

Above: Filter Below:

Commands

Clear Fields Update Search Add Record Delete Record

Employees have full access to stock items as well

Lizzy Lee's - Employee Dashboard

File Edit

Music Item Cafe Item Music Inventory Cafe Inventory

Records

ID	Music Item	Count	Date
1	Warlock	9	14/05/2021
2	SG	10	16/08/2022
3	Explorer	7	11/03/2019
4	SG	7	01/01/2022
5	PRS Custom 24	2	08/12/2020
6	JEM	13	10/02/2022
7	Telecaster	12	23/03/2022
8	PRS Custom 24	3	19/05/2021
9	Les Paul	6	21/11/2020
10	PRS Custom 24	15	05/10/2018
11	SG	6	17/02/2021
12	Les Paul	13	09/11/2022
13	Stratocaster	9	16/10/2021
14	Explorer	5	26/04/2020
15	PRS Custom 24	2	22/10/2018
16	JEM	2	24/11/2018

Stock

Record

ID:  Music Item ID:  Count:   
Date:

Sales

Commands

Clear Fields Search Add Record

But their inventory access is limited to just adding new items.

Lizzy Lee's - Employee Dashboard

File Edit

Music Sales Cafe Sales

Records

ID	Music Item	Customer	Count	Date
1	Flying V	Katie Bevan	3	20/10/2021
2	JEM	Owain Idris	1	04/05/2019
3	JEM	Wyn Roberts	1	11/06/2019
4	SG	Sian Walters	2	01/12/2022
5	Explorer	Elin Roberts	1	22/12/2020
6	Stratocaster	Elin Lloyd	3	26/05/2019
7	Warlock	Huw Morgan	1	24/05/2018
8	Les Paul	Sian Yorath	3	03/10/2019
9	Stratocaster	Owain Price	1	07/02/2020
10	JEM	Ffion Davies	1	19/10/2021
11	Explorer	Huw Morgan	1	26/02/2018
12	JEM	Iwan Ceri	2	21/04/2020
13	Les Paul	Huw Fychan	1	18/08/2019
14	Explorer	Wyn Davies	1	15/10/2020
15	Telecaster	Owain Price	1	16/01/2022
16	PRS Custom 24	Jade Rees	3	03/03/2018

Sales

Record

ID:  Music Item ID:  Customer ID:   
Count:  Date:

Commands

Clear Fields Update Search Add Record

They can also add and update sales, but they can not delete sales that have been recorded

“Teachers” can:

ID	Username	Forename	Surname	Sex	Title	Birthdate	Hire Date	Town	County
1	teacher	Urien	Price	F	Mr	11/03/1994	13/06/2010	Amlwch	Gwynedd

Only view their own record in the “Teacher” tab.

Lizzy Lee's - Teacher Dashboard

**Records**

Teacher	Student									
ID	Username	Forename	Surname	Sex	Title	Birthdate	Town	County	Postcode	Ph
10	Nia799	Mabon	Jenkins	M	Mrs	10/08/1995	Llangefn	Gwynedd	LL77 6BA	0
14	Aneirin865	Branwen	Morgan	F	Mrs	25/03/1993	Beumaris	Conwy	LL77 8YF	0
29	Gethin564	Urien	Roberts	M	Mrs	28/12/1975	Llangefn	Anglesey	LL77 4QU	0
42	Iestyn377	Dyfan	Ap Dafydd	M	Mr	09/01/1993	Beumaris	Anglesey	LL77 4UC	0
43	Haf127	Aneirin	Williams	F	Mrs	11/02/1972	Beumaris	Anglesey	LL77 7JD	0
79	Dyfan297	Haf	Howells	F	Mr	09/02/1981	Gaerwen	Gwynedd	LL77 6LL	0
83	Ffion581	Rhian	Iorwerth	M	Mrs	13/04/1976	Holyhead	Conwy	LL77 0VE	0
87	Haf449	Zara	Edwards	F	Mrs	19/11/1967	Gaerwen	Anglesey	LL77 5QH	0
90	Rhian816	Iestyn	Iorwerth	F	Mr	23/09/1963	Pentraeth	Anglesey	LL77 1BE	0
92	Ynysr195	Branwen	Neville	F	Mr	24/01/1991	Llanfairpwllgwyngyll	Gwynedd	LL77 9TL	0
110	Dyfan268	Mabon	Roberts	M	Mrs	23/04/1960	Llangefn	Conwy	LL77 4DX	0

**Lessons**

**Record**

ID:	<input type="text"/>	Username:	<input type="text"/>	Password:	<input type="text"/>
Forename:	<input type="text"/>	Surname:	<input type="text"/>	Sex:	<input type="text"/>
Title:	<input type="text"/>	Birthdate:	<input type="text"/>	Town:	<input type="text"/>
County:	<input type="text"/>	Postcode:	<input type="text"/>	Phone Number:	<input type="text"/>
Guardian Phone Number:	<input type="text"/>	Email:	<input type="text"/>		

**Range Search**

Birthdate	
After Date:	<input type="text"/> 01/01/1920
Before Date:	<input type="text"/> 01/01/2020
Filter	<input type="button"/>

**Commands**

Clear Fields  Update  Search

They can see and update all the students who they teach as reflected in the “Lesson Bookings” tab.

Lizzy Lee's - Teacher Dashboard

**Records**

ID	Student	Teacher	Location	Lesson Frequency	Lesson Day
21	Zara Edwards	Urien Price	Room 9	Monthly	Tuesday
23	Branwen Morgan	Urien Price	Room 8	Fortnightly	Thursday
47	Iestyn Iorwerth	Urien Price	Room 3	Fortnightly	Friday
49	Dyfan Ap Dafydd	Urien Price	Room 9	Monthly	Monday
67	Mabon Roberts	Urien Price	Room 7	Monthly	Friday
68	Haf Howells	Urien Price	Room 5	Weekly	Wednesday
75	Rhian Iorwerth	Urien Price	Room 9	Weekly	Monday
134	Branwen Neville	Urien Price	Room 4	Fortnightly	Tuesday
148	Mabon Jenkins	Urien Price	Room 9	Weekly	Monday
154	Aneirin Williams	Urien Price	Room 3	Weekly	Wednesday
174	Urien Roberts	Urien Price	Room 8	Fortnightly	Friday

**Lessons**

**Record**

ID:	<input type="text"/>	Student ID:	<input type="text"/>	Teacher ID:	<input type="text"/>
Location:	<input type="text"/>	Lesson Frequency:	<input type="text"/>	Lesson Day:	<input type="text"/>
Lesson Time:	<input type="text"/>	Lesson Length:	<input type="text"/>	Lesson Cost:	<input type="text"/>
Booking Date:	<input type="text"/>	Last Update:	<input type="text"/>	Cancelled:	<input type="text"/>

**Export**

Calculate  Clear  
Export as .txt file:  Export

**Commands**

Clear Fields  Update  Search

Teachers can also see and update their own “Student Bookings” and can also add “Lesson Reports”:

Lizzy Lee's - Teacher Dashboard

**Records**

**Lessons**

**Student Bookings**   **Lesson Report**   **Lesson Plan**

ID	Lesson Title	Student	Teacher	Attended	Student Behaviour
21	Singing Techniques a	Sioned Bevan	Urien Price	0	3
34	Singing Techniques a	Urien Neville	Urien Price	1	4
54	Music Theory for Beg	Cadell Griffiths	Urien Price	1	3
64	Guitar Chords and Te	Ffion Owen	Urien Price	1	4
73	Singing Techniques a	Dyfan Stephens	Urien Price	1	4
89	Guitar Chords and Te	Sioned Bevan	Urien Price	1	0
90	Singing Techniques a	Owain Howells	Urien Price	1	1
93	Guitar Chords and Te	Llio Williams	Urien Price	0	3
120	Songwriting and Con	Zara Neville	Urien Price	0	1
128	Songwriting and Con	Branwen Thomas	Urien Price	0	1
129	Guitar Chords and Te	Wyn Ceredig	Urien Price	0	1
130	Guitar Chords and Te	Zara Howells	Urien Price	1	1
155	Singing Techniques a	Ffion Lloyd	Urien Price	0	2
205	Singing Techniques a	Gethin Edwards	Urien Price	0	1
210	Advanced Drums anc	Dyfan Roberts	Urien Price	1	0
221	Singing Techniques a	Aneirin Price	Urien Price	0	3

**Report Notes**

Note Commands  
Make changes to note:

**Record**

ID:	Lesson Plans ID:	Student ID:
Teacher ID:	Attended:	Student Behaviour:
Notes:	Date:	

**Commands**

Lizzy Lee's - Teacher Dashboard

**Records**

**Lessons**

**Student Bookings**   **Lesson Report**   **Lesson Plan**

ID	Lesson Title	Lesson Objective
1	Guitar Chords and Techniques	Understanding music notation and basic theory co
2	Songwriting and Composition	Mastering complex drum patterns and rhythm
3	Singing Techniques and Performance	Learning basic chords and strumming patterns on
4	Singing Techniques and Performance	Learning basic chords and strumming patterns on
5	Music Theory for Beginners	Mastering complex drum patterns and rhythm
6	Guitar Chords and Techniques	Mastering complex drum patterns and rhythm
7	Guitar Chords and Techniques	Learning basic chords and strumming patterns on
8	Songwriting and Composition	Exploring different songwriting techniques and app
9	Singing Techniques and Performance	Exploring different songwriting techniques and app
10	Singing Techniques and Performance	Learning basic chords and strumming patterns on
11	Music Theory for Beginners	Exploring different songwriting techniques and app
12	Singing Techniques and Performance	Exploring different songwriting techniques and app
13	Guitar Chords and Techniques	Exploring different songwriting techniques and app
14	Music Theory for Beginners	Improving vocal range and control for better singin
15	Songwriting and Composition	Mastering complex drum patterns and rhythm
16	Guitar Chords and Techniques	Understanding music notation and basic theory co

**Lesson procedure**

Procedure Commands  
Make changes to procedure:

**Record**

ID:	Lesson Title:	Lesson Objective:
Materials:	Procedure:	

**Commands**

Teachers can only see “Lesson Plans”. A manager has to add new ones.

Finally, “Students” can:

Lizzy Lee's - Student Dashboard

**File**

**Student**

ID	Username	Forename	Surname	Sex	Title	Birthdate	Town	County	Postcode	Ph
1	student	Gethin	Edwards	F	Mr	13/03/1965	Pentraeth	Conwy	LL77 3FL	0

**Records**

**Lessons**

**Record**

ID:  Username:  Password:   
 Forename:  Surname:  Sex:   
 Title:  Birthdate:  Town:   
 County:  Postcode:  Phone Number:   
 Guardian Phone Number:  Email:

**Range Search**

Birthdate

After Date: 01/01/1920 Before Date: 01/01/2020 Filter

**Commands**

See and edit their own record in the “Students” tab.

And see and edit their own bookings.

Lizzy Lee's - Student Dashboard

**File**

**Student Bookings**

ID	Student	Teacher	Location	Lesson Frequency	Lesson Day
150	Gethin Edwards	Katie Walters	Room 8	Monthly	Tuesday

**Records**

**Lessons**

**Record**

ID:  Student ID:  Teacher ID:   
 Location:  Lesson Frequency:  Lesson Day:   
 Lesson Time:  Lesson Length:  Lesson Cost:   
 Booking Date:  Last Update:  Cancelled:

**Export**

Calculate  Export as .txt file:

**Commands**

Description of intention	What was the issue?	Console Error	Error in code	How I fixed the errors I encountered	Amended code	Date
Tab controller variable created by addressing the Notebook() function	The method for initialising the Tkinter notebook was misspelt	Traceback (most recent call last): File "U:\CA5\Computer Science\Unit5\Unit-5\Practice\main.py", line 35, in <module> app = App() File "U:\CA5\Computer Science\Unit5\Unit-5\Practice\main.py", line 12, in __init__ primaryTabHandler = ttk.notebook(self) AttributeError: 'module' object has no attribute 'notebook'	####Class for handling the application### def __init__(self): super().__init__() self.title("Multi File Classes Testing") menubar = Menu(self) primaryTabHandler = ttk.notebook(self)	Corrected the misspelling of ttk.notebook	####Class for handling the application### def __init__(self): super().__init__() self.title("Multi File Classes Testing") menubar = Menu(self) primaryTabHandler = <b>ttk.Notebook(self)</b>	07/12/2022
Add a command to the menu bar of the Application	Application Menu not adding a command	Traceback (most recent call last): File "U:\CA5\Computer Science\Unit5\Unit-5\Practice\main.py", line 36, in <module> app.add_command(label = "File") File "C:\Python34\lib\ tkinter\_init__.py", line 1932, in __getattr__ return getattr(self.tk, attr) AttributeError: 'tkapp' object has no attribute 'add_command'	app.add_command(label = 'File')	Access the 'add-command' method from the menubar attribute	app.menubar.add_command(label = 'File')	07/12/2022
Creating a menubar for the application	'menubar' attribute couldn't be found	Traceback (most recent call last): File "U:\CA5\Computer Science\Unit5\Unit-5\Practice\main.py", line 36, in <module> app.menubar.add_command(label = "File") File "C:\Python34\lib\ tkinter\_init__.py", line 1932, in __getattr__ return getattr(self.tk, attr) AttributeError: 'tkapp' object has no attribute 'menubar'	menubar = Menu(app)	Menubar must be from 'self', as it is a class attribute	self.menubar = Menu(app)	07/12/2022
Creating a menubar for the application	Couldn't find variable/attribute app	Traceback (most recent call last): File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 55, in <module> app = App() File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 18, in __init__ fileMenu = Menu(app) NameError: name 'app' is not defined. Did you mean: 'App'?	self.menubar = Menu(app)	Called the function Menu() referencing the attribute 'self' instead of 'app' which was instantiated outside the class	self.menubar = Menu( <b>self</b> )	07/12/2022
Add a frame to the main Application	Incorrect option for packing Frame	Traceback (most recent call last): File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 57, in <module> Frame1 = PrimaryTab(app) File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 44, in __init__ self.label.pack(**options) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init_.py", line 2425, in pack_configure self._it_(call) _tkinter.TclError: bad option "-ANCHOR": must be -after, -anchor, -before, -expand, -fill, -in, -ipadx, -ipady, -padx, -pady, or -side	options = {"ANCHOR":W, "fill":True}	Changed 'ANCHOR' to 'anchor'	options = {"anchor":W, "fill":True}	07/12/2022

Add a frame to the main Application	Incorrect option for packing Frame	<pre>Traceback (most recent call last):   File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 57,     in &lt;module&gt;     Frame1 = PrimaryTab(app)   File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 44,     in __init__       self.label.pack(**options)     File   "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 2425, in pack_configure       self.tk.call(     _tkinter.TclError: bad fill style "-1": must be none, x, y, or both</pre>	options = {'anchor':W, 'fill':True}	Changed 'True' to 'both'	options = {'anchor':W, 'fill':'both'}	07/12/2022
Create a Frame that will be added to the applications Notebook tab controller	The subclass PrimaryTab (of Frame) was taking too many arguments	<pre>Traceback (most recent call last):   File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 65,     in &lt;module&gt;       app.createTab("Landing")   File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 36,     in createTab       frame = PrimaryTab(self.primaryTabHandler, name)   File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 44,     in __init__       super().__init__(root, name)     File   "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 3145, in __init__       cnf = _cnfmerge(cnf, kw)     File   "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 111, in _cnfmerge       cnf.update(c) ValueError: dictionary update sequence element #0 has length 1; 2 is required</pre>	class PrimaryTab(Frame):   def __init__(self, root, name):     super().__init__(root, name)	Removed 'name' from the list of inherited arguments	class PrimaryTab(Frame):   def __init__(self, root, name):     super().__init__(root)	07/12/2022
Place the instantiated frame in the notebook	Frame being packed when the manager already has a grid placed item	<pre>Traceback (most recent call last):   File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 59,     in &lt;module&gt;       managerTab = PrimaryTab(app, "Manager")   File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 43,     in __init__       self.pack()     File   "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 2425, in pack_configure       self.tk.call(     _tkinter.TclError: cannot use geometry manager pack inside . which already has slaves managed by grid</pre>	self.pack()	Grid the item instead of packing it	self.grid(row=0, column=0)	07/12/2022
Make dynamic placeholder list for testing Treeview	Placeholder values for testing Treeview using a for loop won't insert variables into the string	<pre>Traceback (most recent call last):   File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 95,     in &lt;module&gt;       employeeTree.insert("", END, values=(str(i), "John"+str(i),       "Smith"+str(i), "Liangrustious", "LL{id}{id}{anti}NY".format(id=str(x),       anti=str(10-x))))</pre>	employeeTree.insert("", END, values=(str(i), "John"+str(i),       "Smith"+str(i), "Liangrustious", "LL{id}{id}{anti}NY".format(id=str(x),       anti=str(10-x))))	Used the f-string type of formatting instead	employeeTree.insert("", END, values=(str(i),       "John"+str(i), "Smith"+str(i), "Liangrustious",       F'LL{id}{id}{anti}NY'))	13/01/2023

Make Frame for buttons have width of 100px	The argument passed was not an option for grid	Traceback (most recent call last): File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 146, in <module> sidebarFrame = Frame(app, width = 100, anchor="ns") File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 3153, in __init__ Widget.__init__(self, master, 'frame', cnf, {}, extra) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 2601, in __init__ self.tk.call('tkinter.TclError: unknown option "-anchor"')	sidebarFrame = Frame(app, width = 100, anchor="ns")	Put the argument "sticky = "ns"" in the grid method	sidebarFrame = Frame(app, width = 100) sidebarFrame.grid(row = 0, column=0, sticky="ns")	19/01/2023
Add an inner padding to a Frame	The argument was passed in the wrong Method	Traceback (most recent call last): File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 205, in <module> employeeTableFrame = Frame(employeeTab, highlightbackground="black", highlightthickness=1, ipady = 5, ipadx = 5) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 3153, in __init__ Widget.__init__(self, master, 'frame', cnf, {}, extra) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 2601, in __init__ self.tk.call('tkinter.TclError: unknown option "-ipady"')	employeeTableFrame = Frame(employeeTab, highlightbackground="black", highlightthickness=1, ipady = 5, ipadx = 5) employeeTableFrame.grid(row = 0, column = 0, sticky ="nw", pady = 10, padx = 10)	The arguments 'ipadx', and 'ipady' should be in the grid method	employeeTableFrame = Frame(employeeTab, highlightbackground="black", highlightthickness=1) employeeTableFrame.grid(row = 0, column = 0, sticky = "nw", pady = 10, padx = 10, ipady = 5, ipadx = 5)	19/01/2023
Add inner padding at only the top of the Frame	Ipady does not accept a tuple for specific directional padding	Traceback (most recent call last): File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 206, in <module> employeeTableFrame.grid(row = 0, column = 0, sticky ="nw", pady = (5, 10), padx = (10, 10), ipady=(3,0)) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 2522, in grid_configure self.tk.call('tkinter.TclError: bad ipady value "3 0": must be positive screen distance')	employeeTableFrame.grid(row = 0, column = 0, sticky ="nw", pady = (5, 10), padx = (10, 10), ipady=(3,0))	Decided against using the inner padding feature	/	19/01/2023
Pass a width and height to the Tk.geometry() method with .format()	.format() was used incorrectly	Traceback (most recent call last): File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 144, in <module> app = App() File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 17, in __init__ self.geometry("{x}").format(width, height) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\__init__.py", line 2073, in wm_geometry return self.tk.call('wm', 'geometry', self._w, newGeometry) _tkinter.TclError: bad geometry specifier "{x}"	self.geometry("{x}").format(width, height)	Place ".format()" at end of string	self.geometry("{x}").format(width, height)	19/01/2023

Insert variable into SQLite query	cursor.execute() not allowing formatting docstring	Traceback (most recent call last): File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 309, in <module> employeeRecord.table.searchQuery() File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 134, in searchQuery queryResults = c.execute("""SELECT * FROM (?)""", (self.table,)) sqlite3.OperationalError: near "?": syntax error	queryResults = c.execute("""SELECT * FROM (?)""", (self.table,))	Make the query a string with string formatting to insert variable	sqlCommand = """SELECT * FROM {}""".format(self.table) queryResults = c.execute(sqlCommand)	19/01/2023
Make dynamic label for each entry in a list	For loop not recognising array as int	Traceback (most recent call last): File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 307, in <module> employeeRecord = Record(employeeTab, "employees") File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 170, in __init__ for x in range(self.table.columns): TypeError: 'list' object cannot be interpreted as an integer	for x in range(self.table.columns):	Use len() to find array's length	for x in range(len(self.table.columns)):	19/01/2023
Set Callback for checking selected tree item	Object method not found	Traceback (most recent call last): File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 309, in <module> employeeRecord = Record(employeeTab, "employees") File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 161, in __init__ self.table = TableFrame(self.tableFrame, table, 620, 350) File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 116, in __init__ self.tree.bind("<ButtonRelease-1>", selectedItem) NameError: name 'selectedItem' is not defined	self.tree.bind("<ButtonRelease-1>", selectedItem)	Call from self	self.tree.bind("<ButtonRelease-1>", self.selectedItem)	20/01/2023
Set Callback for checking selected tree item	Tkinter passing 2 arguments with the callback	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init__.py", line 1921, in __call__ return self.func(*args) TypeError: TableFrame.selectedItem() takes 1 positional argument but 2 were given	self.tree.bind("<ButtonRelease-1>", self.selectedItem)	Add parameter for the selectedItem method	def selectedItem(self, a): self.selection = self.tree.focus() print(self.selection)	20/01/2023
Adjust label text to left of grid	Wrong argument passed	Traceback (most recent call last): File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 315, in <module> employeeRecord = Record(employeeTab, "employees") File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 169, in __init__ Label(self.record, text = str(self.table.columns[x] + ":"), anchor="left").grid(row = row, column = col, padx=(20,0)) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init__.py", line 3177, in __init__ Widget.__init__(self, master, 'label', cnf, kw) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_i	Label(self.record, text = str(self.table.columns[x] + ":"), anchor="left").grid(row = row, column = col, padx=(20,0))	Change argument to 'w'	Label(self.record, text = str(self.table.columns[x] + ":"), anchor="w").grid(row = row, column = col, padx=(20,0))	20/01/2023

		<pre>nit__.py", line 2601, in __init__     self.tk.call(     _tkinter.TclError: bad anchor "left": must be n, ne, e, se, s, sw, w, nw, or center</pre>				
Encrypt data for each element in tables	Python rejecting data as int	<pre>Traceback (most recent call last):   File "C:\CA5\Computer Science\Unit5\Unit-5\Practice\main.py", line 350, in &lt;module&gt;     employeeRecord = Record(employeeTab, "employees")   File "C:\CA5\Computer Science\Unit5\Unit-5\Practice\main.py", line 157, in __init__     self.table = TableFrame(self.tableFrame, table, 620, 350)   File "C:\CA5\Computer Science\Unit5\Unit-5\Practice\main.py", line 128, in __init__     self.searchQuery()   File "C:\CA5\Computer Science\Unit5\Unit-5\Practice\main.py", line 138, in searchQuery     encryptedRow = encrypt(row, 7)   File "C:\CA5\Computer Science\Unit5\Unit-5\Practice\main.py", line 234, in encrypt     for y in x: TypeError: 'int' object is not iterable</pre>	for y in x:	Cast x as a string to prevent being looped through as an int	for y in str(x):	25/01/2023
Remove final comma from string	Operand used isn't supported by string	<pre>Traceback (most recent call last):   File "C:\Users\Wcp2d\AppData\Local\Programs\Python\Python38\lib\tkinter\_it_.py", line 1883, in __call__     return self.func(*args)   File "C:\Users\William\School Work\Computer Science\Unit 5\Unit-5\Practice\main.py", line 232, in searchQuery     query += "," TypeError: unsupported operand type(s) for -: 'str' and 'str'</pre>	query += ","	Use the string as an array with []	query = query[-2]	03/02/2023
Call Update function to update a record's contents	Called wrong method that does not exist	<pre>Traceback (most recent call last):   File "C:\Users\wcp2d\Documents\William\Unit-5\Practice\main.py", line 469, in &lt;module&gt;     employeeRecord = Record(employeeTab, "EMPLOYEE")   File "C:\Users\wcp2d\Documents\William\Unit-5\Practice\main.py", line 206, in __init__     self.updateButton = Button(self.query, text = "Update", command=self.updateQuery) AttributeError: 'Record' object has no attribute 'updateQuery'</pre>	self.updateButton = Button(self.query, text = "Update", command=self.updateQuery)	Rename command	self.updateButton = Button(self.query, text = "Update", command=self.updateRecord)	14/02/2023
Update Row in Table	Incorrect SQL statement	<pre>Exception in Tkinter callback Traceback (most recent call last):   File "C:\Users\wcp2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_it_.py", line 1921, in __call__     return self.func(*args)   File "C:\Users\wcp2d\Documents\William\Unit-5\Practice\main.py", line 282, in updateRecord     c.execute(sql) sqlite3.OperationalError: near "ID": syntax error</pre>	sql = "UPDATE " + self.table.table + " SET "	Add space between UPDATE and SET	sql = "UPDATE " + self.table.table + " SET "	15/02/2023

Update Row in Table	Incorrect SQL statement	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init__.py", line 1921, in __call__ return self.func(*args) File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 282, in updateRecord c.execute(sql) sqlite3.OperationalError: no such column: ClanBill	sql += " WHERE id = " + self.entryVars[self.table.columns[0]].get() + ";"	Put new value in quotation marks	sql += " WHERE id = " + self.entryVars[self.table.columns[0]].get() + ";"	15/02/2023
Set multiple integers to 0	Incorrect syntax	Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init__.py", line 1921, in __call__ return self.func(*args) File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 188, in lambda: self.tree.heading(x, text = str(x), command=lambda x:= self.setHeader(x), anchor= CENTER) File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 231, in headingPushed self.sortDate(column) File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 257, in sortDate year, month, day = 0 TypeError: cannot unpack non-iterable int object App Closed	year, month, day = 0	set each int to zero	year, month, day = 0, 0, 0	15/02/2023
Set Text() widget state to be editable	no state 'enabled'	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init__.py", line 1921, in __call__ return self.func(*args) File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line 666, in calculateCost self.revenueFrame.text["state"] = "enabled" File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init__.py", line 1686, in _setitem__ self.configure(key, value) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init__.py", line 1675, in configure return self._configure(configure, cnf, kw) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init__.py", line 1665, in _configure self.tk.call_(flatten((self._w, cmd)) + self._options(cnf)) _tkinter.TclError: bad state "enabled": must be disabled or normal	self.revenueFrame.text["state"] = "enabled"	change state to "normal"	self.revenueFrame.text["state"] = "normal"	15/02/2023
Create string showing revenue for each row in	Incorrect data types	C:\Users\wcg2d\Documents\William\Unit-5\Practice> c. && cd c:\Users\wcg2d\Documents\William\Unit-5\Practice && cmd /C "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\python.exe c:\Users\wcg2d\vscode\extensions\ms-python.python-2023.2.0\pythonFiles lib\python\debugpy\adapter\..\..\debugpy\launcher 59170 --	self.calculationEstimate += "User ID " + selectedItem[0] + " has " + selectedItem[4] + " monthly lessons, each costing £" + selectedItem[8] + " resulting in a monthly revenue of: £" + monthlyCost + "\n"	cast integers as strings	self.calculationEstimate += "User ID " + str(selectedItem[0]) + " has " + str(selectedItem[4]) + " monthly lessons, each costing £" + str(selectedItem[8]) + " resulting in a monthly revenue of: £" + str(monthlyCost) + "\n"	15/02/2023

a booking		c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py [1, 1, 'Room 1', 1, 'Wednesday', '16:30', 45, 30, '01/02/2023', '09/02/2023', 0] Exception in Tkinter callback Traceback (most recent call last): File "C:/Users/wcg2d/AppData/Local/Programs/Python/Python310/lib/tkinter/_i nit_.py", line 1921, in __call__ return self.func(*args) File "c:/Users/wcg2d/Documents/William[Unit-5]Practice/main.py", line 667, in calculateCost self.calculationEstimate += "User ID " + selectedItem[0] + " has " + selectedItem[4] + " monthly lessons, each costing £" + selectedItem[8] + " resulting in a monthly revenue of: £" + monthlyCost + "\n" TypeError: can only concatenate str (not "int") to str			
Recolour alternating rows after sorting the treeview	tuple being passed instead of int	Exception in Tkinter callback Traceback (most recent call last): File "C:/Users/wcg2d/AppData/Local/Programs/Python/Python310/lib/tkinter/_i nit_.py", line 1921, in __call__ return self.func(*args) File "c:/Users/wcg2d/Documents/William[Unit-5]Practice/main.py", line 195, in <lambda> self.tree.heading(x, text = str(x), command=lambda x=x: self.headingPushed(x), anchor=CENTER) File "c:/Users/wcg2d/Documents/William[Unit-5]Practice/main.py", line 244, in headingPushed self.sortNum(column) File "c:/Users/wcg2d/Documents/William[Unit-5]Practice/main.py", line 332, in sortNum for i in range(0, self.tree.size()): TypeError: 'tuple' object cannot be interpreted as an integer	for i in range(0, self.tree.size()):	use self.tree.getChildren("")	for i in range(0, len(self.tree.get_children(""))):
Call decrypt method in class	Method asking for 2 arguments but only 1 was passed	Traceback (most recent call last): File "c:/Users/wcg2d/Documents/William[Unit-5]Practice/main.py", line 1277, in <module> handler = FileHandler("dummy.log") File "c:/Users/wcg2d/Documents/William[Unit-5]Practice/main.py", line 39, in __init__ self.openFile() File "c:/Users/wcg2d/Documents/William[Unit-5]Practice/main.py", line 51, in openFile self.decryptFile = self.decrypt(Contents) TypeError: FileHandler.decrypt() missing 1 required positional argument: 'privateKey'	def decrypt(self, encryptedMessage, self.privateKey):	Second argument not required	def decrypt(self, encryptedMessage):
Connect an SQL cursor	conn isn't defined as I define it in a class function	Traceback (most recent call last): File "C:/Users/wcg2d/AppData/Local/Programs/Python/Python310/lib/tkinter/_i nit_.py", line 1921, in __call__ return self.func(*args) File "c:/Users/wcg2d/Documents/William[Unit-5]Practice/main.py", line 274, in check_credentials c = conn.cursor()	c = conn.cursor()	Make 'conn' a global variable	global conn
Set tkinter window to	incorrect number of arguments	Traceback (most recent call last): File "c:/Users/wcg2d/Documents/William[Unit-5]Practice/main.py", line	self.wm_attributes('-type', 'splash')	Used 'overridredirect' instead	self.overridredirect(True)

splash screen to remove title bar		<pre>1295, in &lt;module&gt;     splash = Splash(400, 300)   File "c:\Users\wcg2d\Documents\William\Unit-5\Practice\main.py", line   230, in __init__     self.wm_attributes('-type', 'splash')   File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_i   nit_.py", line 2005, in wm_attributes     return self.tk.call(args) _tkinter.TclError: wrong # args: should be "wm attributes window ?alpha ?double?? ?transparentcolor? color?? ?disabled?bool?? ?fullscreen ?bool?? ?toolwindow ?bool?? ?topmost ?bool??"</pre>				
Get File for importing CSV	Incorrect method name	<pre>Exception in Tkinter callback Traceback (most recent call last):   File "C:\Python34\lib\tkinter\_init__.py", line 1533, in __call__     return self.func(*args)   File "U:\CA5\Computer Science\Unit-5-main\PROTOTYPE\main.py", line   458, in importWin     self.selectFile = Button(fileFrame, text = "Select File", command =     self.openFile)   File "C:\Python34\lib\tkinter\_init__.py", line 1932, in __getattr__     return getattr(self.tk, attr) AttributeError: 'kapp' object has no attribute 'openFile'</pre>	<pre>selectFile = Button(fileFrame, text = "Select File", command = self.openFile)</pre>	changed command method name to self.selectFile	<pre>selectFile = Button(fileFrame, text = "Select File", command = self.selectFile)</pre> 01/03/2023	
Set password column of tkinter treeview widget to display asterisks	Couldn't translate code due to repeated 'text' argument.	<pre>C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE&gt; c: &amp;&amp; cd c:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE &amp;&amp; cmd /C "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\python.exe c:\Users\wcg2d\vscode\extensions\ms-python.python-2023.6.0\pythonFiles \lib\python\debugpy\adapter/../debugpy\launcher 5234 - C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py"   File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 682     self.tree.heading(x, text = str(x), command=lambda x=x:     self.headingPushed(x), anchor= CENTER, text = "*****")     ^^^^^^^^^^^^^^ SyntaxError: keyword argument repeated: text</pre>	<pre>self.tree.heading(x, text = str(x), command=lambda x=x: self.headingPushed(x), anchor=CENTER, text = "*****")</pre>	removed initial text argument	<pre>self.tree.heading(x, command=lambda x=x: self.headingPushed(x), anchor=CENTER, text = "*****")</pre> 17/04/2023	
Set password column of tkinter treeview widget to display asterisks	Can't iterate through list of tuples	<pre>Exception in Tkinter callback Traceback (most recent call last):   File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_i   nit_.py", line 1921, in __call__     return self.func(*args)   File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 299, in skip     PrimaryApp(1400, 700)   File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 433, in __init__     employeeRecord = Record(employeeTab, "EMPLOYEE")   File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 1147, in __init__     self.table = TableFrame(self.tableFrame, table, 620, 350)   File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 699, in __init__     self.fillTable()</pre>	for x in range(row):	used len to get the range to loop through	for x in range(len(row)):	17/04/2023

		File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 719, in fillTable for x in range(len(row)): TypeError: 'tuple' object cannot be interpreted as an integer				
Import csv to table	Grabbed the last existing ID from the table, which didn't exist.	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init\_\py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 542, in importCSV idValue = int(last_id) TypeError: int() argument must be a string, a bytes-like object or a real number, not 'NoneType'	idValue = int(last_id)	make an exception for NoneType if no records exist in the table.	if type(last_id) is not int: last_id = 0 idValue = int(last_id)	27/04/2023
Import csv to table	Python can't iterate over type csv.reader	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init\_\py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 553, in importCSV headers = reader[0]	headers = reader[0]	set the csv reader to a list that is iterable	reader = csv.reader(f) # Convert the csv.reader object to a list rows = list(reader)  #Iterate over each row in the file headers = rows[0]	27/04/2023
Import csv to table	Couldn't append text to sql statement string	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init\_\py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 571, in importCSV sql += "VALUES (" + str(idValue) + ", " + " TypeError: bad operand type for unary -: 'str'	sql += " VALUES (" + str(idValue) + ", " + "	changes operand from '+=' to '='	sql = sql + " VALUES (" + str(idValue) + ", " + "	27/04/2023
Import csv to table	SQL statement incorrect	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init\_\py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 579, in importCSV c.execute(sql) sqlite3.OperationalError: near "VALUES": syntax error	for column in headers: sql += column + ", " sql = sql[:-2]	add closing bracket to sql statement	for column in headers: sql += column + ", " sql = sql[:-2] + ")"	27/04/2023
Check employee role on login	Can't find table 'employee'	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init\_\py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 349, in checkCredentials	permission = c.execute("SELECT job_description FROM employee WHERE username = ? ", (username,)).fetchone()[0]	change casing of table name to EMPLOYEE	permission = c.execute("SELECT job_description FROM EMPLOYEE WHERE username = ? ", (username,)).fetchone()[0]	27/04/2023

		permission = c.execute('SELECT job_description FROM employees WHERE username = ?', (username)).fetchone()[0] sqlite3.OperationalError: no such table: employees				
Check employee role on login	Can't execute sql statement	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init_.py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 349, in checkCredentials permission = c.execute(str("SELECT job_description FROM EMPLOYEE WHERE username = " + username)),fetchone() sqlite3.OperationalError: no such column: LizLee!	permission = c.execute(str("SELECT job_description FROM EMPLOYEE WHERE username = " + username)).fetchone()	add commas around the username value entered to the statement	permission = c.execute(str("SELECT job_description FROM EMPLOYEE WHERE username = " + username + "")).fetchone()	27/04/2023
Use debug button to skip login	App couldn't open main window	Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init_.py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 308, in skip PrimaryApp(1400, 700, user_role = "Manager") File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 447, in __init__ employeeTab = PrimaryTab(self, recordTabs, "Employee") TypeError: PrimaryTab.__init__() missing 1 required positional argument: 'profile'	def __init__(self, root, tabHandler, name, profile) -> None:	removed unused parameter "profile"	def __init__(self, root, tabHandler, name) -> None:	03/05/2023
Bind password entry to show it's field when clicked on	Typo in code	Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init_.py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 308, in skip PrimaryApp(1400, 700, user_role = "Manager") File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 453, in __init__ employeeRecord = Record(employeeTab, "EMPLOYEE") File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 1205, in __init__ self.entries[len(self.entries) - 1].bind('<FocusIn>', lambda event: entry.config(show="")) NameError: name 'ser' is not defined	self.entries[len(self.entries) - 1].bind('<FocusIn>', lambda event: entry.config(show=""))	corrected spelling	self.entries[len(self.entries) - 1].bind('<FocusIn>', lambda event: entry.config(show=""))	03/05/2023
Bind password entry to show it's field when clicked on	Incorrect variable name for binding	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init_.py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 1205, in __lambda__ self.entries[len(self.entries) - 1].bind('<FocusIn>', lambda event: entry.config(show="")) NameError: name 'entry' is not defined	self.entries[len(self.entries) - 1].bind('<FocusIn>', lambda event: entry.config(show=""))	change "entry" variable to "self.entries[len(self.entries) - 1]"	self.entries[len(self.entries) - 1].bind('<FocusIn>', lambda event: self.entries[len(self.entries) - 1].config(show=""))	03/05/2023

Set width of 'password' column in treeview to zero	Multiple 'width' parameters in class instantiation	File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 722 self.tree.column(x, anchor=CENTER, minWidth=30, width=120, stretch=False, width = 0) ^^^^^^^ SyntaxError: keyword argument repeated: width	self.tree.column(x, anchor=CENTER, minWidth=30, width=120, stretch=False, width = 0)	Remove duplicate width parameter	self.tree.column(x, anchor=CENTER, minWidth=0, width=0, stretch=False)	03/05/2023
Import table into lesson booking	Unique constraint in SQL not satisfied for Lesson Report ID	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init\_\py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 618, in importCSV c.execute(sql) sqlite3.IntegrityError: UNIQUE constraint failed: LESSON_REPORT.ID	N/A	Changed import file to not specify ID's. This way the system will handle ID's itself to avoid duplicate ID's from being imported	N/A	04/05/2023
Set layout of widgets for lesson report tab	Python couldn't find a variable called noteFrame	Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init\_\py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 320, in skipPrimaryApp PrimaryApp(1400, 700, user_role = "Manager") File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 495, in __init__ lessonReportFrame = Report(lessonReportTab, "LESSON REPORT") File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 1499, in __init__ self.noteText = TextFrame(noteFrame, width=500, height = 300) NameError: name 'noteFrame' is not defined	<pre>self.displayNotes = LabelFrame(noteFrame, text = "Note Commands")</pre>	changed variable "noteFrame" to "self.noteFrame" to reference class attribute	<pre>self.displayNotes = LabelFrame(self.noteFrame, text = "Note Commands")</pre>	04/05/2023
Execute SQL statement to update note column of selected record	Python wouldn't append the record ID to SQL statement string, because it is an int	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init\_\py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 1538, in saveNote sqlQuery = "UPDATE " + self.table.table + " SET Notes = " + self.selection[6] + " WHERE ID = " + self.selection[0] TypeError: can only concatenate str (not "int") to str	sqlQuery = "UPDATE " + self.table.table + " SET Notes = " + self.selection[6] + " WHERE ID = " + self.selection[0]	cast the ID as a string using "str()"	sqlQuery = "UPDATE " + self.table.table + " SET Notes = " + self.selection[6] + " WHERE ID = " + str(self.selection[0])	05/05/2023
Execute SQL statement to update note column of selected record	SQL didn't understand the parameter in the SQL statement. It needs to have quotation marks surrounding it. Additionally the wrong variable is	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\tkinter\_init\_\py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 1540, in saveNote c.execute(sqlQuery)	sqlQuery = "UPDATE " + self.table.table + " SET Notes = " + self.selection[6] + " WHERE ID = " + str(self.selection[0])	Add quotation marks surrounding the parameter, and change the variable	sqlQuery = "UPDATE " + self.table.table + " SET Notes = \\" + self.noteText.text.get(1.0, END)	05/05/2023

	being passed - the current "Notes" value is being passed instead of the new one	sqlite3.OperationalError: unrecognized token: "2gXA53VspgRNuvHcktRQZ0oq3Pg4rqZYW"		+ "\\" WHERE ID = " + str(self.selection[0])		
Execute SQL code to create Lesson Plan table	SQL code missing closing bracket	Traceback (most recent call last): File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 165, in <module> handler = FileHandler("savedata.lz") File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 30, in __init__ self.createTables() File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 207, in createTables c.execute("") sqlite3.OperationalError: incomplete input	c.execute("""         CREATE TABLE IF NOT EXISTS LESSON_PLAN (             ID INTEGER PRIMARY KEY             AUTOINCREMENT,             LessonTitle TEXT(50),             lessonObjective TEXT(50000),             materials TEXT(50000),             procedure TEXT(50000)         )""")	Add closing bracket to end of statement	c.execute("""         CREATE TABLE IF NOT EXISTS LESSON_PLAN (             ID INTEGER PRIMARY KEY             AUTOINCREMENT,             LessonTitle TEXT(50),             lessonObjective TEXT(50000),             materials TEXT(50000),             procedure TEXT(50000)         )""")	06/05/2023
Make backup file in "Backup" Folder	Filename passed does not match the time format because the file path was passed	Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\ tkinter\__init__.py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 572, in onExit handler.closeFile() File "C:\Users\wcg2d\Documents\William\Unit-5\PROTOTYPE\main.py", line 65, in closeFile file_date = datetime.strptime(os.path.splitext(file_path)[0], "%d-%m-%Y") File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\_strptime.py", line 568, in _strptime_datetime tt, fraction, gmtoff, fraction = _strptime(data_string, format) File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\_strptime.py", line 349, in _strptime raise ValueError("time data %s does not match format %s" % ValueError: time data 'backups\06-05-2023' does not match format "%d-%m-%Y")	file_date = datetime.strptime(os.path.splitext(file_path)[0], "%d-%m-%Y")	splice the variable passed to only include the date	file_date = datetime.strptime(os.path.splitext(file_path)[0], "%d-%m-%Y")	06/05/2023
Login to Employee Account	Application passing the table into titleCase() function couldn't split the parameter input because the account logged into had no defined	Exception in Tkinter callback Traceback (most recent call last): File "C:\Users\wcg2d\AppData\Local\Programs\Python\Python310\lib\ tkinter\__init__.py", line 1921, in __call__ return self.func(*args) File "C:\Users\wcg2d\Documents\William\Unit-5\FINAL\main.py", line 405, in checkCredentials PrimaryApp(1400, 700, user_role = titleCase(permission))	permission = c.execute(str("SELECT job_description FROM EMPLOYEE WHERE password = '" + password + "'")) .fetchone () [0]	Ensured record had job_description	N/A	08/05/2023



```
self.noteText.text.delete(1.0,  
END)  
  
self.noteText.text.insert(END,  
self.selection[6])  
  
self.noteText.text['state'] =  
"disabled"
```

# Testing

## TEST PLAN

To ensure all aspects of the final product will be functional to the end user, extensive testing will be carried out covering all aspects of the program to ensure no issues will occur during its (hopefully) long lifetime.

The tester must perform numerous checks for every input the system has to offer, and additionally checks must be made to ensure that system-specific processes function properly under various use cases (such as ensuring data from an import is validated properly)

The following is a plan designed for a tester to use to test every aspect of the program. The testing will be carried out after the program is completed.

# CALCULATIONS:

## Lesson Bookings Monthly Estimate:

"Student Bookings" tab tests regarding the calculations results from selected records:

Section: Lessons		Tab: Student Bookings		
Test Number	Purpose of Test	Data to Test	Expected outcome	Additional Test Notes
1	Ensure calculations are accurate for monthly estimates with frequency set to weekly	lesson_frequency = Weekly, lesson_cost = 30, cancelled = No	120	
2	Ensure calculations are accurate for monthly estimates with frequency set to fortnightly	lesson_frequency = Fortnightly, lesson_cost = 30, cancelled = No	60	
3	Ensure calculations are accurate for monthly estimates set to monthly	lesson_frequency = Monthly, lesson_cost = 30, cancelled = No	30	
4	Ensure calculations account for if the booking has been cancelled	lesson_frequency = Monthly, lesson_cost = 30, cancelled = Yes	0	
5	Ensure the system can calculate total revenue from multiple bookings containing	lesson_frequency = Monthly, lesson_cost = 30, cancelled = No AND ANOTHER RECORD WHERE lesson_frequency	110	

	different values	= Weekly, lesson_cost = 20, cancelled = No		
6	Ensure the system can calculate total revenue from multiple bookings with one cancelled	lesson_frequency = Monthly, lesson_cost = 30, cancelled = Yes AND ANOTHER RECORD WHERE lesson_frequency = Weekly, lesson_cost = 20, cancelled = No		80
7	Ensure the system can deal with no bookings being selected	No selection		A system prompt should inform the user 0 to select a row

### Cafe Inventory Value Estimate:

“Cafe Inventory” tab tests to ensure calculations of the cafe inventory are correct:

Section: Stock		Tab: Cafe Inventory		
Test Number	Purpose of Test	Data to Test	Expected outcome	Additional Notes
1	Ensure system's inventory value estimate is correct	Select one record in the Cafe Inventory table where count = 3 and links to a Cafe Item with value of £2.25	6.75	

		Select one record in the Cafe Inventory table where count = 3 and links to a Cafe Item with value of £2.25 And another record where count = 2 and links to a Cafe Item with value of £1.20		
2	Ensure system's inventory value estimate is correct for multiple records		9.15	

## Music Inventory Value Estimate:

“Music Inventory” tab tests to ensure calculations of the music inventory are correct:

Section: Stock		Tab: Music Inventory		
Test Number	Purpose of Test	Data to Test	Expected outcome	Additional Notes
1	Ensure system's inventory value estimate is correct	Select one record in the Music Inventory table where count = 2 and links to a Music Item with value of £89	178	
2	Ensure system's inventory value estimate is correct for multiple records	Select one record in the Music Inventory table where count = 2 and links to a Music Item with value of £89 And another record where count = 3 and links to a Music Item with value of £12	214	

**Cafe Sales Estimate:**

"Cafe Sales" tab tests to ensure calculations of the cafe sales income are correct:

1	Ensure system's Sales value estimate is correct	Select one record in the Cafe Sales table where count = 3 and links to a Cafe Item with value of £2.25	6.75	
2	Ensure system's Sales value estimate is correct for multiple records	Select one record in the Cafe Sales table where count = 3 and links to a Cafe Item with value of £2.25 And another record where count = 2 and links to a Cafe Item with value of £1.20	9.15	

**Music Sales Estimate:**

"Music Sales" tab tests to ensure calculations of the music sales income are correct:

Section: Sales		Tab: Music Sales		
Test Number	Purpose of Test	Data to Test	Expected outcome	Additional Notes
1	Ensure system's Sales value estimate is correct	Select one record in the Music Sales table where count = 2 and links to a Music Item with value of £89	178	

		Select one record in the Music Sales table where count = 2 and links to a Music Item with value of £89 And another record where count = 3 and links to a Music Item with value of £12		
2	Ensure system's Sales value estimate is correct for multiple records		214	

## VALIDATIONS

Any validation techniques that are reused will not need to be tested. This happens in several places such as for Town and County, or Forename and Surname and such.

Section: Records				Tab: Employee	
Test Num ber	Field to Test	Field Input	Test Purpose	Significance of Input Choice	Expected outcome
1	Username	ClainBill	Normal Data check Test should be okay	First Letter: Uppercase letter Type: only alphabetical letters Length: 9 satisfies $3 \geq \text{length} \geq 25$	Accepts Username
2	Username	dog	Lower Bound check	First Letter: Lowercase letter Type: only lowercase letters	Accepts Username

			Uses only one type of data and minimum length allowed	Length: 3 satisfies $3 \geq \text{length} \geq 25$	
3	Username	Abcd3fhhijklmnopqrstuvwxyz!	Upper Bound check Uses all non-whitespace data types and is 25 characters long	First Letter: Uppercase letter Type: All non-whitespace data types used at least once Length: 25 satisfies $3 \geq \text{length} \geq 25$	Accepts Username
4	Username	!aUsername	Invalid Data check Should be invalid	First Letter: Symbol Type: N/A Length: N/A	Rejects Username for starting with symbol
5	Username	ab	Invalid Data check Should be invalid	First Letter: N/A Type: N/A Length: 1 does not satisfy $3 \geq \text{length} \geq 25$	Rejects Username for being too short
6	Username	abcdefghijklmnopqrstuvwxyz	Invalid Data check Should be invalid	First Letter: N/A Type: N/A Length: 26 does not satisfy $3 \geq \text{length} \geq 25$	Rejects Username for being too long
7	Password	Passw0rd!	Normal Data check Test should be okay	Type: Contains at least 1 of every character type	Accepts Password

				Length: 9 satisfies $8 \geq \text{length} \geq 30$	
8	Password	Wrkpl3@s	Lower Bound check Uses only one of each data type and minimum length allowed	Type: Minimum number of types needed Length 8 satisfies $8 \geq \text{length} \geq 30$	Accepts Password
9	Password	Rc4l!fragilisticexpialidocious	Upper Bound check Uses all non-whitespace data types and is 30 characters long	Type: All types needed Length 30 satisfies $8 \geq \text{length} \geq 30$	Accepts Password
10	Password	Password	Invalid Data check Should be invalid	Type: Does not contain every data type needed	Rejects Password Needs at least 1 of all input types
11	Password	E^!1y	Invalid Data check Should be invalid	Length 5 does not satisfy $8 \geq \text{length} \geq 30$	Rejects Password Needs to be between 8 and 30 chars long
12	Forename	William	Normal Data check Test should be okay	First Letter: Uppercase letter Type: only alphabetical letters Length: 9 satisfies $3 \geq \text{length} \geq 25$	Accepts Forename

13	Forename	Bob	Lower Bound check Uses only one of each data type and minimum length allowed	First Letter: Uppercase letter Type: only alphabetical letters Length: 3 satisfies $3 \geq \text{length} \geq 25$	Accepts Forename
14	Forename	Christophermichaelsonsson	Upper Bound check Uses all non-whitespace data types and is 25 characters long	First Letter: Uppercase letter Type: only alphabetical letters Length: 9 satisfies $3 \geq \text{length} \geq 25$	Accepts Forename
15	Forename	timmy	Invalid Data check Should be invalid	First Letter: Lowercase letter Type: only lowercase letters Length: 5 satisfies $3 \geq \text{length} \geq 25$	Rejects Forename Must begin with Uppercase letter and proceed with only lowercase letters
16	Forename	AndrEw	Invalid Data check Should be invalid	Type: Has uppercase letter after first letter	Rejects Forename Must begin with Uppercase letter and proceed with only lowercase letters
17	Forename	Ja	Invalid Data check Should be invalid	Length: 2 does not satisfy $3 \geq \text{length} \geq 25$	Rejects Forename Must begin with Uppercase letter and

					proceed with only lowercase letters
18	Sex	M	Normal Data check Test should be okay	M is in lookup check for sex	Accepts sex field
19	Sex	Mississippi River	Invalid Data check Should be invalid	"Mississippi River" is not in lookup check for sex	Rejects sex field Must be either 'M' or 'F'
20	Birthdate	09/04/2005	Normal Data check Test should be okay	Type: Only numbers and '/' Format: Correctly formatted date Range: Is a date in the past	Accepts Birthdate
21	Birthdate	09/05/2023	Upper Bound check Is yesterday's date	Type: Only numbers and '/' Format: Correctly formatted date Range: Is the latest date still in the past	Accepts Birthdate
22	Birthdate	00/00/0000	Lower Bound check Is the earliest date representable in this format	Type: Only numbers and '/' Format: Correctly formatted date Range: Is earliest date possible	Accepts Birthdate
23	Birthdate	1/01/1998	Invalid Data check Should be invalid	Type: Only numbers and '/' Format: Incorrectly formatted date	Rejects Birthdate

				Range: Is a date in the past	
24	Birthdate	01-01-1998	Invalid Data check Should be invalid	Type: Only numbers and '/' Format: Incorrectly formatted date Range: Is a date in the past	Rejects Birthdate
25	Birthdate	first of january 1998	Invalid Data check Should be invalid	Type: Has alphabetical characters Format: Incorrectly formatted date	Rejects Birthdate
26	Birthdate	09/05/2025	Invalid Data check Should be invalid	Type: Only numbers and '/' Format: Correctly formatted date Range: Is a date in the future	Rejects Birthdate
27	Town	Llangefni	Normal Data check Test should be okay	Type: Only letters Format: Starts with capital letter	Accepts Town
28	Town	Winchester-on-the-Severn	Normal Data check Test should be okay	Type: Only letters and hyphens Format: Starts with capital letter	Accepts Town
29	Town	Menai Bridge	Normal Data check Test should be okay	Type: Only letters and spaces Format: Starts with capital letters	Accepts Town
30	Town	Water 7	Invalid Data check Should be invalid	Type: Contains a number	Rejects Town

31	Town	tOkyO	Invalid Data check Should be invalid	Format: Does not start with capital letter	Rejects Town
32	Postcode	LL77 7GH	Normal Data check Test should be okay	Format: Follows correct postcode format of LL00 0LL	Accepts Postcode
33	Postcode	LL777GH	Invalid Data check Should be invalid	Format: Does not follow correct postcode format of LL00 0LL	Rejects Postcode
34	Postcode	0LHS 81S	Invalid Data check Should be invalid	Format: Does not follow correct postcode format of LL00 0LL	Rejects Postcode
35	Phone Number	0726274201	Normal Data check Test should be okay	Type: Contains only numbers Format: Follows UK phone number format	Accepts Phone Number
36	Phone Number	07TwoSix4201	Invalid Data check Should be invalid	Type: Contains Letters Format: Does not follow UK phone number format	Rejects Phone Number
37	Phone Number	620074160575	Invalid Data check Should be invalid	Format: Does not follow UK phone number format	Rejects Phone Number

38	Email	william@freaks.org.uk	Normal Data check Test should be okay	Format: Follows conventional email format	Accepts Email
39	Email	william.freaks.org.uk	Invalid Data check Should be invalid	Format: Doesn't have @ symbol	Rejects Email
40	Email	!william@freaks.org.uk	Invalid Data check Should be invalid	Type: Has non-@ symbol	Rejects Email

Section: Stock				Tab: Cafe Item	
Test Number	Field to Test	Field Input	Test Purpose	Significance of Input Choice	Expected outcome
1	Price	12.5	Normal Data check Test should be okay	Type: Float Range: Within 0-99999	Accepts Price
2	Price	0	Lower Bound check Uses minimum value allowed	Type: Float Range: Within 0-99999	Accepts Price

3	Price	99999	Upper Bound check Uses maximum value allowed	Type: Float Range: Within 0-99999	Accepts Price
4	Price	-1	Invalid Data check Should be invalid	Type: Float Range: Not within 0-99999	Rejects Price Must be in range
5	Price	twelve	Invalid Data check Should be invalid	Type: String	Rejects Price Must be a number
6	Price	100000	Invalid Data check Exceeds maximum value allowed	Range: Above 0-99999	Rejects Price Must be in range

Section: Stock				Tab: Cafe Inventory	
Test Number	Field to Test	Field Input	Test Purpose	Significance of Input Choice	Expected outcome

1	Count	3	Normal Data check Test should be okay	Type: Integer Range: Within 0-9999	Accepts Price
2	Count	0	Lower Bound check Uses minimum value allowed	Type: Integer Range: Within 0-9999	Accepts Price
3	Count	9999	Upper Bound check Uses maximum value allowed	Type: Integer Range: Within 0-9999	Accepts Price
4	Count	-1	Invalid Data check Should be invalid	Range: Not within 0-9999	Rejects Price Must be in range
5	Count	twelve	Invalid Data check Should be invalid	Type: String	Rejects Price Must be a number
6	Count	10000	Invalid Data check Exceeds maximum value allowed	Range: Above 0-9999	Rejects Price Must be in range

7	Count	3.5	Invalid Data check Should be invalid	Type: Not an integer	Rejects Price Must be in range

## OPERATIONS TESTING

The various operations the system can provide need to be tested to ensure they work.

The following tests ensure the login system functions correctly

Window: Login				
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes
1	Ensure system will restrict incorrect credentials	Enter "admin" in the username field. Enter "Password" in the password field. Click the "Login" button or press Enter	A popup will tell the user their credentials were incorrect	
2	Ensure system will allow correct credentials	Enter "admin" in the username field. Enter "admin" in the password field. Click the "Login" button or press Enter	The system allows the user and gives them access to the "Manager Dashboard" as noted in the title of the window	

3	Ensure the system will provide employees with a different level access to managers	Enter "employee" in the username field. Enter "Passw0rd!" in the password field. Click the "Login" button or press Enter	The system allows the user and gives them access to the "Employee Dashboard" as noted in the title of the window	"employee" is an example employee record I have preloaded in the system with these credentials
4	Ensure the system will provide teachers with a different level access to managers	Enter "teacher" in the username field. Enter "Passw0rd!" in the password field. Click the "Login" button or press Enter	The system allows the user and gives them access to the "Teacher Dashboard" as noted in the title of the window	"teacher" is an example employee record I have preloaded in the system with these credentials
5	Ensure the system will provide students with a different level access to managers	Enter "student" in the username field. Enter "Passw0rd!" in the password field. Click the "Login" button or press Enter	The system allows the user and gives them access to the "Student Dashboard" as noted in the title of the window	"student" is an example employee record I have preloaded in the system with these credentials

These records work identically in all tabs. The only difference is different users might not have access to all commands in every tab

Section: Records	Tab: Employees
------------------	----------------

<b>Test Number</b>	<b>Purpose of Test</b>	<b>Instruction</b>	<b>Expected outcome</b>	<b>Additional Test Notes</b>
1	Ensure Clear Entries Button Works	Click on "Clear Fields" button when all fields are filled with data	All entries in this record are cleared	
2	Ensure user can Add Records	Fill in all entries and click "Add Record" button	A popup confirms the record was added and the new record appears in the treeview grid	
3	Ensure user can Edit Records	Click on a record and change any entry other than the ID Click on the "Update" button	A popup confirms the record was updated and the updated record appears in the treeview grid	
4	Ensure user can Delete Records	Click on a record and click on the "Delete" button. Click "OK" on the popup that will appear	A popup asks the user to confirm the user wants to delete the record. Then when confirmed the record is removed from the treeview and the entries are cleared	

5	Ensure user can Search Records with one search parameter	Type a value into any record field that would make sense to appear (such as 'M' in the 'Sex' field) Click on the "Search" button	The treeview should update to only show results that have that exact data in the specified column	
6	Ensure user can Search Records with multiple search parameters	Type another value into any other record field that would make sense to appear (such as 'Llangefni' in the 'Town' field) Click on the "Search" button	The treeview should update to only show results that have that exact data in all the specified columns	If no results appear adjust the filters to be more common
7	Ensure the user can perform "After" date range filters	With no entries filled (push the "Clear Fields" button) put a date in the "After Date" field of the birthdate range check in the "Range Check" frame on the right. Click on "Filter"	The treeview should filter out any results that had a birthdate before the date entered	If no results appear try adjusting the range
8	Ensure the user can perform "Before" and "After" date range filters	With no entries filled (push the "Clear Fields" button) put a date in the "After Date" and "Before Date" field of the birthdate range check in the "Range Check" frame on the right. Click on "Filter"	The treeview should filter out any results that had a birthdate outside the range given	

9	Ensure the user can perform date range filters to their current search	With the same filter applied, apply a "Search" filter by typing a sensible value into a record field. Click on "Filter" inside the "Range Check" frame on the right with the filled in range	The treeview should display only results that match all of the filters applied	If no results appear try adjusting the search and filter parameters to be more broad

Toolbar Menu at top of window				
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes
1	Ensure Autosave works	Make a change to a record (such as updating, adding or deleting a record) Hold down the "Alt" key on the keyboard and then simultaneously press the "F4" key to force close the program.  Reopen the program and the changes should be saved.	When the program is opened again the changes made before force closing the program should be maintained.	
2	Ensure "Logout" command works	Click on "File" and then hover over and select "Logout" from the drop down menu.	The program should save and close the main program window, and the login window should automatically open	
3	Ensure "Exit" command works	Click on "File" and then hover over and select "Exit"	The program should save	

		from the drop down menu.	and close the main program window	
4	Ensure "Import" command works	Click on "Edit" and then hover over and select "Import" from the drop down menu.	A new import window should appear above the primary application allowing.	The "Edit" dropdown menu is only accessible by the Managers and Employees. So MAKE SURE YOU'RE LOGGED IN AS SUCH
5	Ensure "Export" command works	Click on "Edit" and then hover over and select "Export" from the drop down menu.	A new export window should appear above the primary application allowing.	The "Edit" dropdown menu is only accessible by the Managers and Employees. So MAKE SURE YOU'RE LOGGED IN AS SUCH

Import Window				
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes

			A popup says success and tells the user to refresh the table to see results. If the test was successful the imported record should be seen at the end of the "Employees" table when the "Search" button is pressed with no entries filled	username,password,forename,surname,birthdate,job_descrip tion newUser, newPassw0rd!, Timothy, Tester, 01/01/1980, employee
1	Ensure importing works correctly	Create a '.csv' file that contains the text in the Additional Notes of this test.  Select the "Employee" table in the selection box.  Click the "Import" button and select the file you created in the file selection window that appears.	A popup says the file was invalid	username ,password newUser, newPassw0rd!, Timothy, Tester, 01/01/1980, employee

Export Window				
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes
1	Ensure exporting works correctly	Select any table in the selection box.  Click the "Export" button and select a file location	A file is created in the location specified	

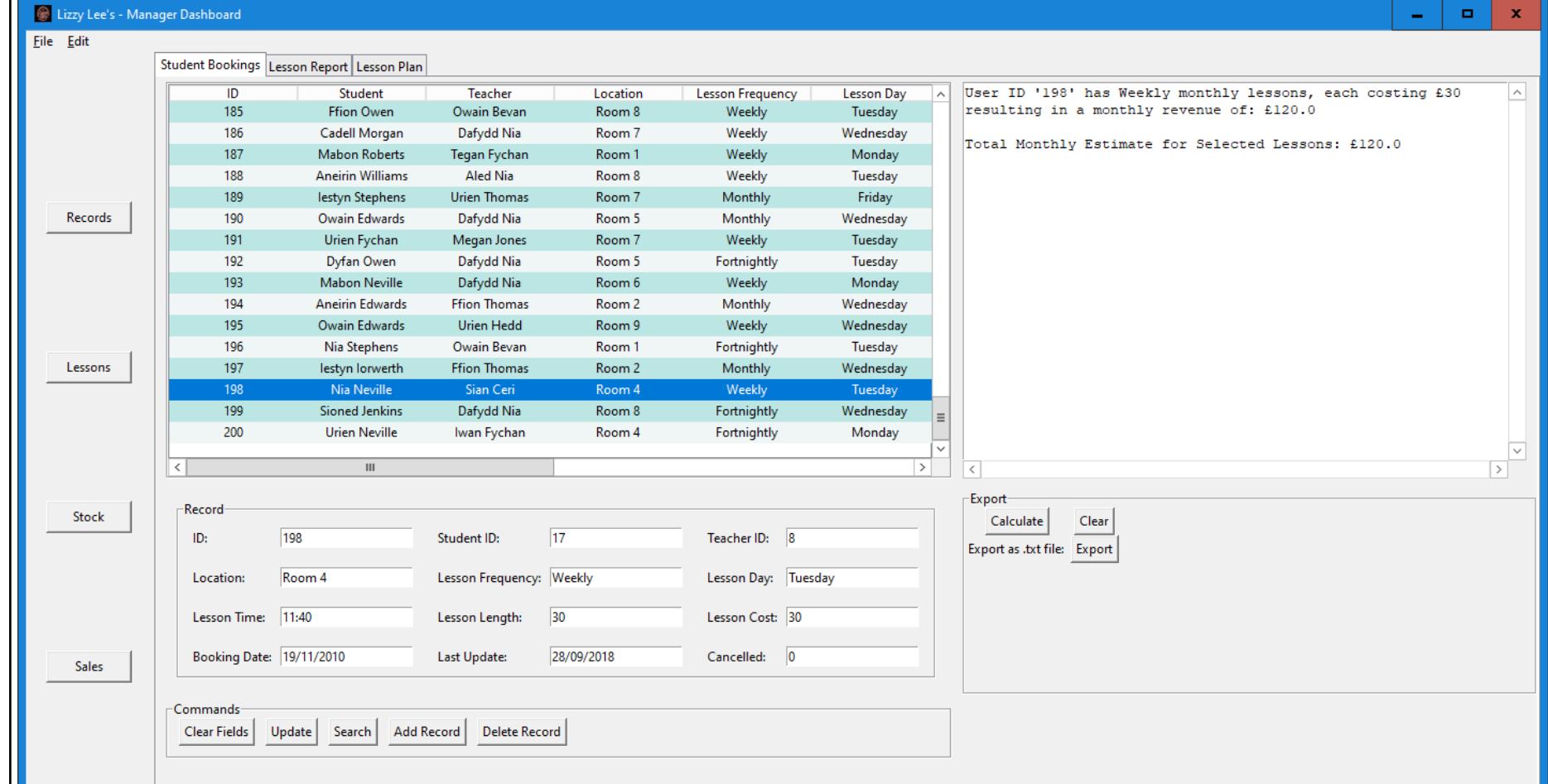
Section: Lessons		Tab: Student Bookings		
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes
1	Ensure calculations are successfully exported	Select either one or multiple records in the treeview by shift-clicking. Click on the "Calculate" button in the "Export" frame on the right. Then click on "Export"	A "LessonEstimate.txt" file is created in the directory of the application	
2	Ensure calculations are successfully added to the existing file	Again, Select either one or multiple records in the treeview by shift-clicking. Click on the "Calculate" button in the "Export" frame on the right. Then click on "Export"	The "LessonEstimate.txt" file is appended to with the new calculation made	

# EVIDENCE OF TEST RESULTS

## Lesson Bookings Monthly Estimate:

“Student Bookings” tab tests regarding the calculations results from selected records:

Section: Lessons		Tab: Student Bookings		
Test Number	Purpose of Test	Data to Test	Expected outcome	Additional Test Notes
1	Ensure calculations are accurate for monthly estimates with frequency set to weekly	lesson_frequency = Weekly, lesson_cost = 30, cancelled = No	120	

Section: Lessons		Tab: Student Bookings	
 <p>The screenshot shows a software application window titled "Lizzy Lee's - Manager Dashboard". The main content area is a "Student Bookings" grid with columns: ID, Student, Teacher, Location, Lesson Frequency, and Lesson Day. The grid contains 20 records. Record 198 is highlighted in blue. To the right of the grid, there is a message box displaying calculated results: "User ID '198' has Weekly monthly lessons, each costing £30 resulting in a monthly revenue of: £120.0" and "Total Monthly Estimate for Selected Lessons: £120.0". Below the grid is a "Record" form with fields for ID (198), Student ID (17), Teacher ID (8), Location (Room 4), Lesson Frequency (Weekly), Lesson Day (Tuesday), Lesson Time (11:40), Lesson Length (30), Lesson Cost (30), Booking Date (19/11/2010), Last Update (28/09/2018), and Cancelled (0). At the bottom left is a "Commands" section with buttons for Clear Fields, Update, Search, Add Record, and Delete Record. On the left side of the dashboard, there are four buttons: Records, Lessons, Stock, and Sales.</p>			

The system has passed the test successfully. The record selected has a Lesson Frequency of Weekly, and a lesson cost of £30. When clicking the "Calculate" button the program returns the expected result of £120

Section: Lessons		Tab: Student Bookings		
2	Ensure calculations are accurate for monthly estimates with frequency set to fortnightly	lesson_frequency = Fortnightly, lesson_cost = 30, cancelled = No	60	

Lizzy Lee's - Manager Dashboard

File Edit

Records Lessons Stock Sales

Student Bookings Lesson Report Lesson Plan

ID	Student	Teacher	Location	Lesson Frequency	Lesson Day
185	Ffion Owen	Owain Bevan	Room 8	Weekly	Tuesday
186	Cadell Morgan	Dafydd Nia	Room 7	Weekly	Wednesday
187	Mabon Roberts	Tegan Fychan	Room 1	Weekly	Monday
188	Aneirin Williams	Aled Nia	Room 8	Weekly	Tuesday
189	Iestyn Stephens	Urien Thomas	Room 7	Monthly	Friday
190	Owain Edwards	Dafydd Nia	Room 5	Monthly	Wednesday
191	Urien Fychan	Megan Jones	Room 7	Weekly	Tuesday
192	Dyfan Owen	Dafydd Nia	Room 5	Fortnightly	Tuesday
193	Mabon Nevill	Dafydd Nia	Room 6	Weekly	Monday
194	Aneirin Edwards	Ffion Thomas	Room 2	Monthly	Wednesday
195	Owain Edwards	Urien Hedd	Room 9	Weekly	Wednesday
196	Nia Stephens	Owain Bevan	Room 1	Fortnightly	Tuesday
197	Iestyn Iorwerth	Ffion Thomas	Room 2	Monthly	Wednesday
198	Nia Neville	Sian Ceri	Room 4	Fortnightly	Tuesday
199	Sioned Jenkins	Dafydd Nia	Room 8	Fortnightly	Wednesday
200	Urien Neville	Iwan Fychan	Room 4	Fortnightly	Monday

User ID '198' has Fortnightly monthly lessons, each costing £30 resulting in a monthly revenue of: £60.0

Total Monthly Estimate for Selected Lessons: £60.0

Record

ID:	198	Student ID:	17	Teacher ID:	8
Location:	Room 4	Lesson Frequency:	Fortnightly	Lesson Day:	Tuesday
Lesson Time:	11:40	Lesson Length:	30	Lesson Cost:	30
Booking Date:	19/11/2010	Last Update:	28/09/2018	Cancelled:	0

Commands

Clear Fields Update Search Add Record Delete Record

Export

Calculate Clear Export as .txt file: Export

The system passes the test with the expected result of £60

Section: Lessons		Tab: Student Bookings		
3	Ensure calculations are accurate for monthly estimates set to monthly	lesson_frequency = Monthly, lesson_cost = 30, cancelled = No	30	

Lizzy Lee's - Manager Dashboard

File Edit

Records Lessons Stock Sales

**Student Bookings** Lesson Report Lesson Plan

ID	Student	Teacher	Location	Lesson Frequency	Lesson Day
185	Ffion Owen	Owain Bevan	Room 8	Weekly	Tuesday
186	Cadell Morgan	Dafydd Nia	Room 7	Weekly	Wednesday
187	Mabon Roberts	Tegan Fychan	Room 1	Weekly	Monday
188	Aneirin Williams	Aled Nia	Room 8	Weekly	Tuesday
189	Iestyn Stephens	Urien Thomas	Room 7	Monthly	Friday
190	Owain Edwards	Dafydd Nia	Room 5	Monthly	Wednesday
191	Urien Fychan	Megan Jones	Room 7	Weekly	Tuesday
192	Dyfan Owen	Dafydd Nia	Room 5	Fortnightly	Tuesday
193	Mabon Nevill	Dafydd Nia	Room 6	Weekly	Monday
194	Aneirin Edwards	Ffion Thomas	Room 2	Monthly	Wednesday
195	Owain Edwards	Urien Hedd	Room 9	Weekly	Wednesday
196	Nia Stephens	Owain Bevan	Room 1	Fortnightly	Tuesday
197	Iestyn Iorwerth	Ffion Thomas	Room 2	Monthly	Wednesday
198	Nia Neville	Sian Ceri	Room 4	Monthly	Tuesday
199	Sioned Jenkins	Dafydd Nia	Room 8	Fortnightly	Wednesday
200	Urien Neville	Iwan Fychan	Room 4	Fortnightly	Monday

User ID '198' has Monthly monthly lessons, each costing £30 resulting in a monthly revenue of: £30.0

Total Monthly Estimate for Selected Lessons: £30.0

Record

ID:	198	Student ID:	17	Teacher ID:	8
Location:	Room 4	Lesson Frequency:	Monthly	Lesson Day:	Tuesday
Lesson Time:	11:40	Lesson Length:	30	Lesson Cost:	30
Booking Date:	19/11/2010	Last Update:	28/09/2018	Cancelled:	0

Commands

Clear Fields Update Search Add Record Delete Record

Export

Calculate Clear Export as .txt file: Export

The system passes the test by returning a value of £30

Section: Lessons	Tab: Student Bookings
4 Ensure calculations account for if the booking has been cancelled	lesson_frequency = Monthly, lesson_cost = 30, cancelled = Yes 0

Lizzy Lee's - Manager Dashboard

File Edit

Records Lessons Stock Sales

Student Bookings Lesson Report Lesson Plan

Lesson Time	Lesson Length	Lesson Cost	Booking Date	Last Update	Cancelled
15:27	30	35	15/04/2011	24/03/2020	1
08:26	60	35	04/10/2017	23/12/2015	0
10:04	60	25	11/03/2017	03/03/2011	0
14:16	30	35	09/01/2016	04/01/2014	0
15:57	45	20	05/05/2016	14/10/2018	0
10:01	30	20	14/08/2020	10/09/2019	0
16:53	60	30	19/05/2013	21/02/2014	0
09:21	30	30	19/03/2016	02/02/2014	0
09:17	30	35	08/07/2013	25/04/2013	0
12:37	60	40	11/07/2013	10/12/2013	1
14:08	30	20	20/12/2012	22/04/2010	1
10:06	30	20	10/12/2020	13/11/2011	0
14:49	30	20	04/05/2016	05/07/2011	1
11:40	30	30	19/11/2010	28/09/2018	1
11:55	30	30	22/05/2014	24/04/2019	1
08:00	60	25	23/11/2014	15/03/2014	1

Total Monthly Estimate for Selected Lessons: £0

Record

ID:	198	Student ID:	17	Teacher ID:	8
Location:	Room 4	Lesson Frequency:	Monthly	Lesson Day:	Tuesday
Lesson Time:	11:40	Lesson Length:	30	Lesson Cost:	30
Booking Date:	19/11/2010	Last Update:	28/09/2018	Cancelled:	1

Commands

Clear Fields Update Search Add Record Delete Record

Export

Calculate Clear Export as .txt file: Export

The system passes the test. The monthly estimate is £0 as was the expected outcome

Section: Lessons	Tab: Student Bookings		
5 Ensure the system can calculate total revenue from multiple bookings containing 5 different values	lesson_frequency = Monthly, lesson_cost = 30, cancelled = No AND ANOTHER RECORD WHERE lesson_frequency = Weekly, lesson_cost = 20, cancelled = No	110	

Section: Lessons      Tab: Student Bookings

Lizzy Lee's - Manager Dashboard

**Records**

**Lessons**

**Stock**

**Sales**

**Student Bookings**   **Lesson Report**   **Lesson Plan**

ID	Student	Teacher	Location	Lesson Frequency	Lesson Day
185	Ffion Owen	Owain Bevan	Room 8	Weekly	Tuesday
186	Cadell Morgan	Dafydd Nia	Room 7	Weekly	Wednesday
187	Mabon Roberts	Tegan Fychan	Room 1	Weekly	Monday
188	Aneirin Williams	Aled Nia	Room 8	Weekly	Tuesday
189	Iestyn Stephens	Urien Thomas	Room 7	Monthly	Friday
190	Owain Edwards	Dafydd Nia	Room 5	Monthly	Wednesday
191	Urien Fychan	Megan Jones	Room 7	Weekly	Tuesday
192	Dyfan Owen	Dafydd Nia	Room 5	Fortnightly	Tuesday
193	Mabon Nevill	Dafydd Nia	Room 6	Weekly	Monday
194	Aneirin Edwards	Ffion Thomas	Room 2	Monthly	Wednesday
195	Owain Edwards	Urien Hedd	Room 9	Weekly	Wednesday
196	Nia Stephens	Owain Bevan	Room 1	Fortnightly	Tuesday
197	Iestyn Iorwerth	Ffion Thomas	Room 2	Monthly	Wednesday
198	Nia Neville	Sian Ceri	Room 4	Monthly	Tuesday
199	Sioned Jenkins	Dafydd Nia	Room 8	Weekly	Wednesday
200	Urien Neville	Iwan Fychan	Room 4	Fortnightly	Monday

User ID '198' has Monthly monthly lessons, each costing £30 resulting in a monthly revenue of: £30.0  
User ID '199' has Weekly monthly lessons, each costing £20 resulting in a monthly revenue of: £80.0  
Total Monthly Estimate for Selected Lessons: £110.0

**Record**

ID:	199	Student ID:	72	Teacher ID:	4
Location:	Room 8	Lesson Frequency:	Weekly	Lesson Day:	Wednesday
Lesson Time:	11:55	Lesson Length:	30	Lesson Cost:	20
Booking Date:	22/05/2014	Last Update:	24/04/2019	Cancelled:	0

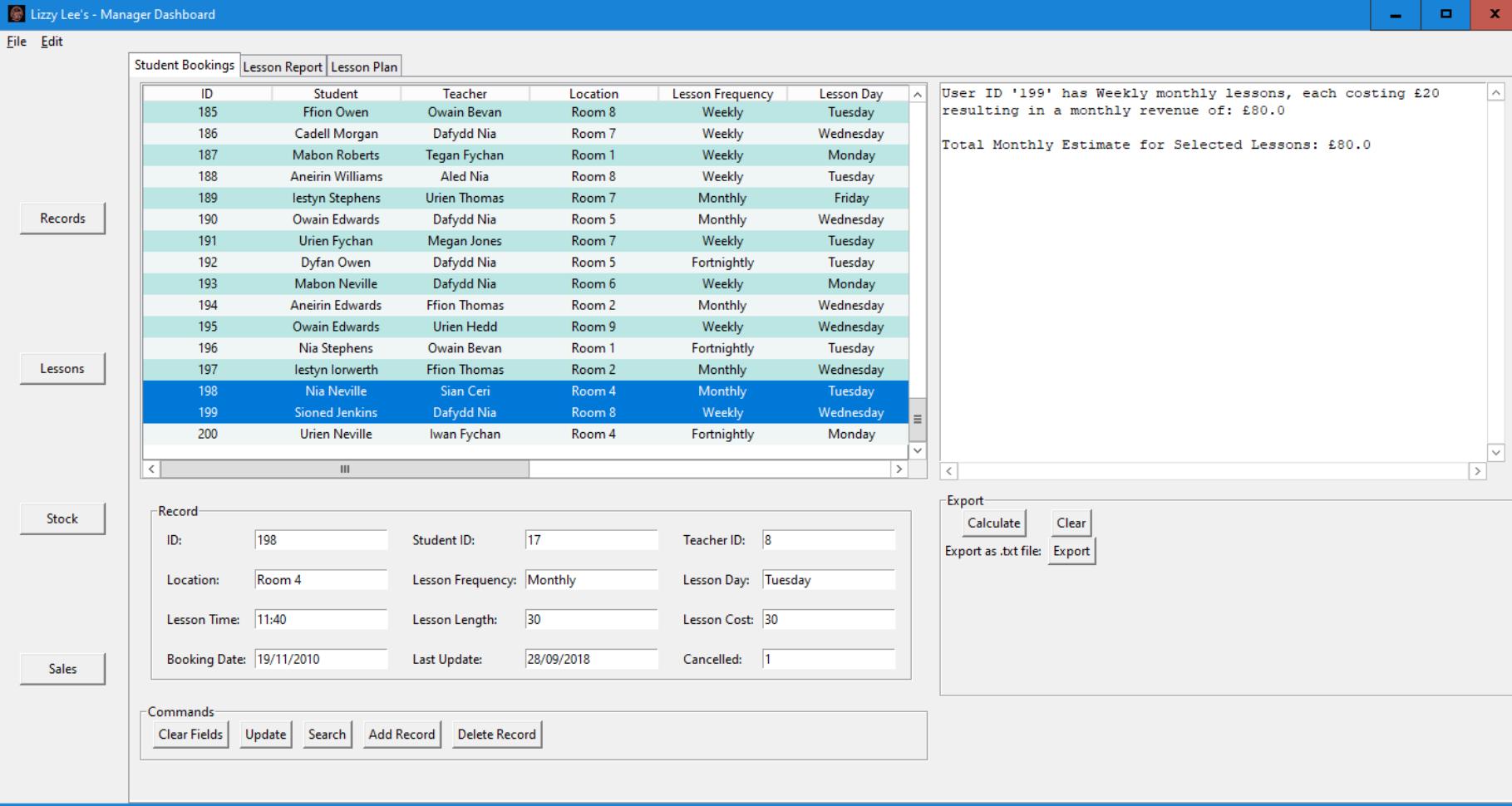
**Commands**

**Export**

Calculate   Clear  
Export as .txt file:

The system passes the test. The first record returns a monthly revenue of £30, and the second adds a monthly revenue of £80, totalling the expected £110

Section: Lessons	Tab: Student Bookings		
6	Ensure the system can calculate total revenue from multiple bookings with one cancelled	lesson_frequency = Monthly, lesson_cost = 30, cancelled = Yes AND ANOTHER RECORD WHERE lesson_frequency = Weekly, lesson_cost = 20, cancelled = No	80

Section: Lessons		Tab: Student Bookings																																																																																																							
 <p>The screenshot shows a software application window titled "Lizzy Lee's - Manager Dashboard". The main area is a grid titled "Student Bookings" showing lesson details:</p> <table border="1"> <thead> <tr> <th>ID</th> <th>Student</th> <th>Teacher</th> <th>Location</th> <th>Lesson Frequency</th> <th>Lesson Day</th> </tr> </thead> <tbody> <tr><td>185</td><td>Ffion Owen</td><td>Owain Bevan</td><td>Room 8</td><td>Weekly</td><td>Tuesday</td></tr> <tr><td>186</td><td>Cadell Morgan</td><td>Dafydd Nia</td><td>Room 7</td><td>Weekly</td><td>Wednesday</td></tr> <tr><td>187</td><td>Mabon Roberts</td><td>Tegan Fychan</td><td>Room 1</td><td>Weekly</td><td>Monday</td></tr> <tr><td>188</td><td>Aneirin Williams</td><td>Aled Nia</td><td>Room 8</td><td>Weekly</td><td>Tuesday</td></tr> <tr><td>189</td><td>Iestyn Stephens</td><td>Urien Thomas</td><td>Room 7</td><td>Monthly</td><td>Friday</td></tr> <tr><td>190</td><td>Owain Edwards</td><td>Dafydd Nia</td><td>Room 5</td><td>Monthly</td><td>Wednesday</td></tr> <tr><td>191</td><td>Urien Fychan</td><td>Megan Jones</td><td>Room 7</td><td>Weekly</td><td>Tuesday</td></tr> <tr><td>192</td><td>Dyfan Owen</td><td>Dafydd Nia</td><td>Room 5</td><td>Fortnightly</td><td>Tuesday</td></tr> <tr><td>193</td><td>Mabon Nevill</td><td>Dafydd Nia</td><td>Room 6</td><td>Weekly</td><td>Monday</td></tr> <tr><td>194</td><td>Aneirin Edwards</td><td>Ffion Thomas</td><td>Room 2</td><td>Monthly</td><td>Wednesday</td></tr> <tr><td>195</td><td>Owain Edwards</td><td>Urien Hedd</td><td>Room 9</td><td>Weekly</td><td>Wednesday</td></tr> <tr><td>196</td><td>Nia Stephens</td><td>Owain Bevan</td><td>Room 1</td><td>Fortnightly</td><td>Tuesday</td></tr> <tr><td>197</td><td>Iestyn Iorwerth</td><td>Ffion Thomas</td><td>Room 2</td><td>Monthly</td><td>Wednesday</td></tr> <tr><td>198</td><td>Nia Neville</td><td>Sian Ceri</td><td>Room 4</td><td>Monthly</td><td>Tuesday</td></tr> <tr><td>199</td><td>Sioned Jenkins</td><td>Dafydd Nia</td><td>Room 8</td><td>Weekly</td><td>Wednesday</td></tr> <tr><td>200</td><td>Urien Neville</td><td>Iwan Fychan</td><td>Room 4</td><td>Fortnightly</td><td>Monday</td></tr> </tbody> </table> <p>On the right side of the dashboard, there are two text boxes with calculated results:</p> <ul style="list-style-type: none"> <li>User ID '199' has Weekly monthly lessons, each costing £20 resulting in a monthly revenue of: £80.0</li> <li>Total Monthly Estimate for Selected Lessons: £80.0</li> </ul> <p>On the left sidebar, there are four categories: Records, Lessons, Stock, and Sales. The "Lessons" category is currently selected.</p>				ID	Student	Teacher	Location	Lesson Frequency	Lesson Day	185	Ffion Owen	Owain Bevan	Room 8	Weekly	Tuesday	186	Cadell Morgan	Dafydd Nia	Room 7	Weekly	Wednesday	187	Mabon Roberts	Tegan Fychan	Room 1	Weekly	Monday	188	Aneirin Williams	Aled Nia	Room 8	Weekly	Tuesday	189	Iestyn Stephens	Urien Thomas	Room 7	Monthly	Friday	190	Owain Edwards	Dafydd Nia	Room 5	Monthly	Wednesday	191	Urien Fychan	Megan Jones	Room 7	Weekly	Tuesday	192	Dyfan Owen	Dafydd Nia	Room 5	Fortnightly	Tuesday	193	Mabon Nevill	Dafydd Nia	Room 6	Weekly	Monday	194	Aneirin Edwards	Ffion Thomas	Room 2	Monthly	Wednesday	195	Owain Edwards	Urien Hedd	Room 9	Weekly	Wednesday	196	Nia Stephens	Owain Bevan	Room 1	Fortnightly	Tuesday	197	Iestyn Iorwerth	Ffion Thomas	Room 2	Monthly	Wednesday	198	Nia Neville	Sian Ceri	Room 4	Monthly	Tuesday	199	Sioned Jenkins	Dafydd Nia	Room 8	Weekly	Wednesday	200	Urien Neville	Iwan Fychan	Room 4	Fortnightly	Monday
ID	Student	Teacher	Location	Lesson Frequency	Lesson Day																																																																																																				
185	Ffion Owen	Owain Bevan	Room 8	Weekly	Tuesday																																																																																																				
186	Cadell Morgan	Dafydd Nia	Room 7	Weekly	Wednesday																																																																																																				
187	Mabon Roberts	Tegan Fychan	Room 1	Weekly	Monday																																																																																																				
188	Aneirin Williams	Aled Nia	Room 8	Weekly	Tuesday																																																																																																				
189	Iestyn Stephens	Urien Thomas	Room 7	Monthly	Friday																																																																																																				
190	Owain Edwards	Dafydd Nia	Room 5	Monthly	Wednesday																																																																																																				
191	Urien Fychan	Megan Jones	Room 7	Weekly	Tuesday																																																																																																				
192	Dyfan Owen	Dafydd Nia	Room 5	Fortnightly	Tuesday																																																																																																				
193	Mabon Nevill	Dafydd Nia	Room 6	Weekly	Monday																																																																																																				
194	Aneirin Edwards	Ffion Thomas	Room 2	Monthly	Wednesday																																																																																																				
195	Owain Edwards	Urien Hedd	Room 9	Weekly	Wednesday																																																																																																				
196	Nia Stephens	Owain Bevan	Room 1	Fortnightly	Tuesday																																																																																																				
197	Iestyn Iorwerth	Ffion Thomas	Room 2	Monthly	Wednesday																																																																																																				
198	Nia Neville	Sian Ceri	Room 4	Monthly	Tuesday																																																																																																				
199	Sioned Jenkins	Dafydd Nia	Room 8	Weekly	Wednesday																																																																																																				
200	Urien Neville	Iwan Fychan	Room 4	Fortnightly	Monday																																																																																																				
<p>The test passes the test. The total monthly revenue of both lessons is estimated at £80 as was expected</p>																																																																																																									

Section: Lessons	Tab: Student Bookings		
7 Ensure the system can deal with no bookings being selected	No selection	0	A system prompt should inform the user to select a row

Section: Lessons      Tab: Student Bookings

ID	Student	Teacher	Location	Lesson Frequency	Lesson Day
1	Llio Stephens	Dafydd Nia	Room 8	Monthly	Tuesday
2	Dyfan Ceredig	Katie Roberts	Room 7	Fortnightly	Tuesday
3	Dyfan Fychan	Owain Bevan	Room 3	Fortnightly	Friday
4	Owain Howells	Gareth Idris	Room 7	Monthly	Tuesday
5	Haf Lloyd	Iwan Yorath	Room 9	Weekly	Wednesday
6	Llio Khan	Iwan Fychan	Room 1	Fortnightly	Friday
7	Cadell Owen	Katie Walters	Room 3	Monthly	Thursday
8	Rhian Davies	Dafydd Nia	Room 2	Fortnightly	Thursday
9	Dyfan Williams	Aled Edwards	Room 5	Weekly	Monday
10	Yn yr Bevan	Iwan Yorath	Room 6	Fortnightly	Tuesday
11	Dyfan Owen	Iwan Price	Room 2	Weekly	Tuesday
12	Dyfan Ap Dafydd	Gareth Idris	Room 10		
13	Mabon Khan	Iwan Yorath	Room 4		
14	Nia Thomas	Iwan Price	Room 10		
15	Mabon Khan	Iwan Fychan	Room 6		
16	Dyfan Khan	Megan Jones	Room 6		
17	Iestvn Iorwerth	Tegan Fychan	Room 4		

The system responds as expected. A warning prompt tells the user to select a row

## Cafe Inventory Value Estimate:

“Cafe Inventory” tab tests to ensure calculations of the cafe inventory are correct:

Section: Stock		Tab: Cafe Inventory												
Test Number	Purpose of Test	Data to Test	Expected outcome	Additional Notes										
1	Ensure system's inventory value estimate is correct	Select one record in the Cafe Inventory table where count = 3 and links to a Cafe Item with value of £2.25	6.75											
The Cafe Item has a value of £2.25														
<table border="1"> <thead> <tr> <th>ID</th> <th>Type</th> <th>Description</th> <th>Price</th> <th>Expiry Date</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Cookie</td> <td>Thumbprint Cookie</td> <td>2.25</td> <td>12/09/2029</td> </tr> </tbody> </table>					ID	Type	Description	Price	Expiry Date	1	Cookie	Thumbprint Cookie	2.25	12/09/2029
ID	Type	Description	Price	Expiry Date										
1	Cookie	Thumbprint Cookie	2.25	12/09/2029										

The screenshot shows a software application window with two main sections. On the left is a grid-based data entry or display screen with four columns: ID, Cafe Item, Count, and Date. The data consists of 17 rows of items. On the right is a vertical panel titled 'Inventory Estimate' containing a single line of text: 'Estimated inventory value: £6.75'. The entire window is enclosed in a black border.

ID	Cafe Item	Count	Date
1	Thumbprint Cookie	3	07/12/2019
2	Chocolate Chai	10	09/06/2018
3	Teh Halia	2	09/03/2019
4	Diet Coke	4	08/09/2020
5	A&W Root Beer	3	07/12/2019
6	Pain	5	13/03/2020
7	Diet Coke Lemon	8	11/05/2018
8	Biscotti	11	18/04/2022
9	Cafe Con Leche	8	03/03/2021
10	Pain au Lard	7	26/01/2020
11	Caffe Mocha	2	08/04/2020
12	German Chocolate Ci	11	24/03/2021
13	Biscotti	9	07/01/2018
14	Diet Coke Cherry	15	14/06/2022
15	Diet Coke Raspberry	11	23/06/2020
16	Pain au Lard	8	06/09/2020
17	Tuna Sandwich	10	26/10/2018

When the inventory record linking to that item is selected the Inventory estimate correctly outputs £6.75

	Ensure system's inventory value estimate is 2 correct for multiple records	Select one record in the Cafe Inventory table where count = 3 and links to a Cafe Item with value of £2.25 And another record where count = 2 and links to a Cafe Item with value of £1.20		9.15
--	---	---	--	------

The two cafe items with the specified prices:

ID	Type	Description	Price	Expiry Date
1	Cookie	Thumbprint Cookie	2.25	12/09/2029
2	Pastry	Pain au Raisin	1.2	17/06/2027

ID	Cafe Item	Count	Date
1	Thumbprint Cookie	3	07/12/2019
2	Pain au Raisin	2	09/06/2018
3	Teh Halia	2	09/03/2019
4	Diet Coke	4	08/09/2020
5	A&W Root Beer	3	07/12/2019
6	Pain	5	13/03/2020
7	Diet Coke Lemon	8	11/05/2018
8	Biscotti	11	18/04/2022
9	Cafe Con Leche	8	03/03/2021
10	Pain au Lard	7	26/01/2020
11	Caffe Mocha	2	08/04/2020
12	German Chocolate Ci	11	24/03/2021
13	Biscotti	9	07/01/2018
14	Diet Coke Cherry	15	14/06/2022
15	Diet Coke Raspberry	11	23/06/2020
16	Pain au Lard	8	06/09/2020
17	Tuna Sandwich	10	26/10/2018

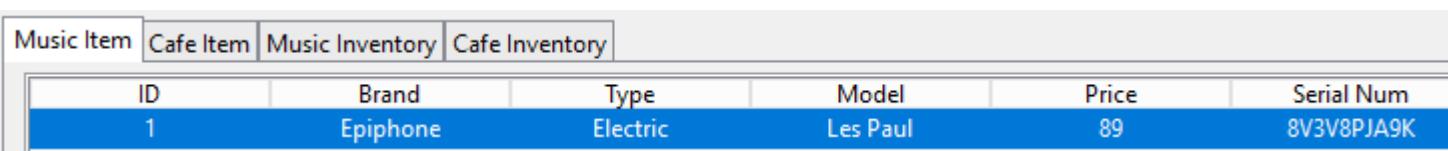
Inventory Estimate

Estimated inventory value: £9.15

The records corresponding to the separate cafe items return an estimated inventory value of £9.15

## Music Inventory Value Estimate:

“Music Inventory” tab tests to ensure calculations of the music inventory are correct:

Section: Stock		Tab: Music Inventory																		
Test Number	Purpose of Test	Data to Test	Expected outcome	Additional Notes																
1	Ensure system's inventory value estimate is correct	Select one record in the Music Inventory table where count = 2 and links to a Music Item with value of £89	178																	
 <table border="1"> <thead> <tr> <th>Music Item</th><th>Cafe Item</th><th>Music Inventory</th><th>Cafe Inventory</th></tr> </thead> <tbody> <tr> <td>ID</td><td>Brand</td><td>Type</td><td>Model</td><td>Price</td><td>Serial Num</td></tr> <tr style="background-color: #0070C0; color: white;"> <td>1</td><td>Epiphone</td><td>Electric</td><td>Les Paul</td><td>89</td><td>8V3V8PJA9K</td></tr> </tbody> </table>					Music Item	Cafe Item	Music Inventory	Cafe Inventory	ID	Brand	Type	Model	Price	Serial Num	1	Epiphone	Electric	Les Paul	89	8V3V8PJA9K
Music Item	Cafe Item	Music Inventory	Cafe Inventory																	
ID	Brand	Type	Model	Price	Serial Num															
1	Epiphone	Electric	Les Paul	89	8V3V8PJA9K															

The Music Inventory record associated with the Music Item of worth £89 correctly returns an estimate inventory value of £178

Music Item	Cafe Item	Music Inventory	Cafe Inventory
			Inventory Estimate
			Estimated inventory value: £178
ID	Music Item	Count	Date
1	Les Paul	2	14/05/2021
2	SG	10	16/08/2022
3	Explorer	7	11/03/2019
4	SG	7	01/01/2022
5	PRS Custom 24	2	08/12/2020
6	JEM	13	10/02/2022
7	Telecaster	12	23/03/2022
8	PRS Custom 24	3	19/05/2021
9	Les Paul	6	21/11/2020
10	PRS Custom 24	15	05/10/2018
11	SG	6	17/02/2021
12	Les Paul	13	09/11/2022
13	Stratocaster	9	16/10/2021
14	Explorer	5	26/04/2020
15	PRS Custom 24	2	22/10/2018
16	JEM	2	24/11/2018
17	JEM	11	26/08/2021

	Select one record in the Music Inventory table where count = 2 and links to a Music Item with value of £89 And another record where count = 3 and links to a Music Item with value of £12		
2	Ensure system's inventory value estimate is correct for multiple records	214	

Music Item	Cafe Item	Music Inventory	Cafe Inventory
ID	Brand	Type	Model
1	Epiphone	Electric	Les Paul
2	Schecter	Classical	JEM
			Price
			Serial Num
			8V3V8PJA9K
			7YDP6MMS7P

Two music items of worth £89 and £12 respectively

The Inventory correctly returns a value of £214

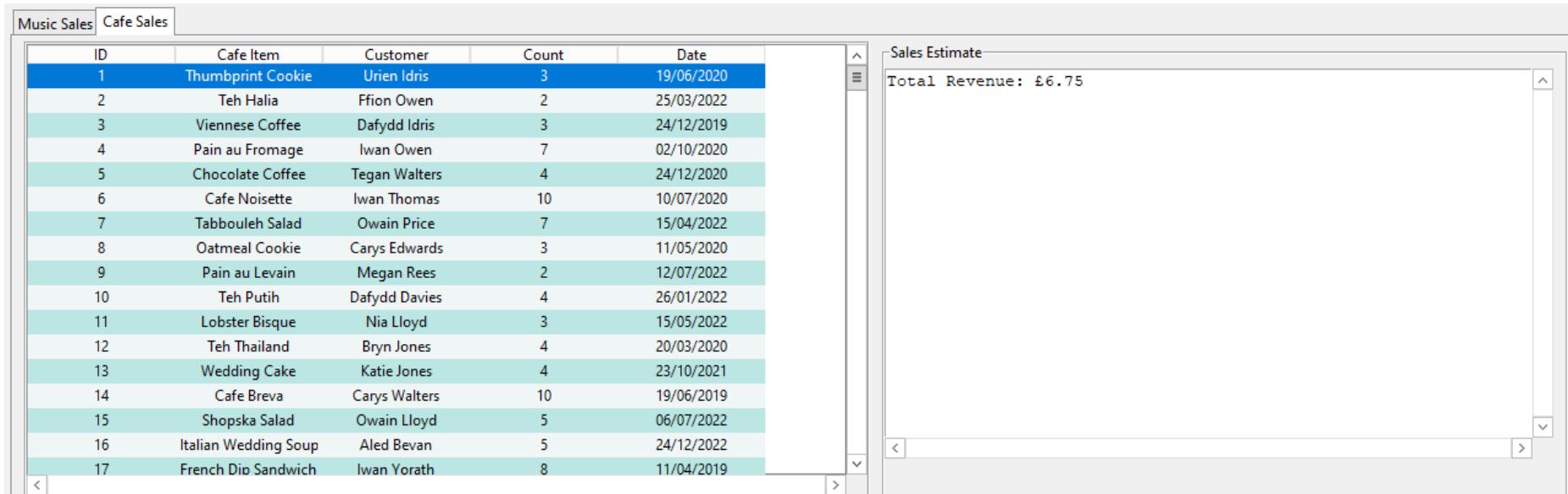
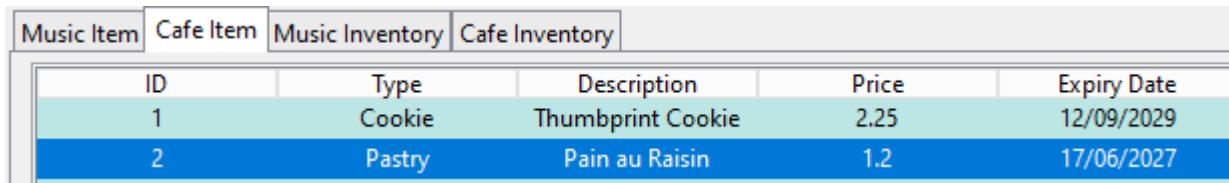
Music Item	Cafe Item	Music Inventory	Cafe Inventory
1	Les Paul	2	14/05/2021
2	JEM	3	16/08/2022
3	Explorer	7	11/03/2019
4	SG	7	01/01/2022
5	PRS Custom 24	2	08/12/2020
6	JEM	13	10/02/2022
7	Telecaster	12	23/03/2022
8	PRS Custom 24	3	19/05/2021
9	Les Paul	6	21/11/2020
10	PRS Custom 24	15	05/10/2018
11	SG	6	17/02/2021
12	Les Paul	13	09/11/2022
13	Stratocaster	9	16/10/2021
14	Explorer	5	26/04/2020
15	PRS Custom 24	2	22/10/2018
16	JEM	2	24/11/2018
17	JEM	11	26/08/2021

Inventory Estimate  
Estimated inventory value: £214

### Cafe Sales Estimate:

“Cafe Sales” tab tests to ensure calculations of the cafe sales income are correct:

1	Ensure system's Sales value estimate is correct	Select one record in the Cafe Sales table where count = 3 and links to a Cafe Item with value of £2.25	6.75															
<table border="1"> <thead> <tr> <th>Music Item</th><th>Cafe Item</th><th>Music Inventory</th><th>Cafe Inventory</th></tr> </thead> <tbody> <tr><td>ID</td><td>Type</td><td>Description</td><td>Price</td><td>Expiry Date</td></tr> <tr><td>1</td><td>Cookie</td><td>Thumbprint Cookie</td><td>2.25</td><td>12/09/2029</td></tr> </tbody> </table> <p>Cafe Item of worth 2.25</p>					Music Item	Cafe Item	Music Inventory	Cafe Inventory	ID	Type	Description	Price	Expiry Date	1	Cookie	Thumbprint Cookie	2.25	12/09/2029
Music Item	Cafe Item	Music Inventory	Cafe Inventory															
ID	Type	Description	Price	Expiry Date														
1	Cookie	Thumbprint Cookie	2.25	12/09/2029														

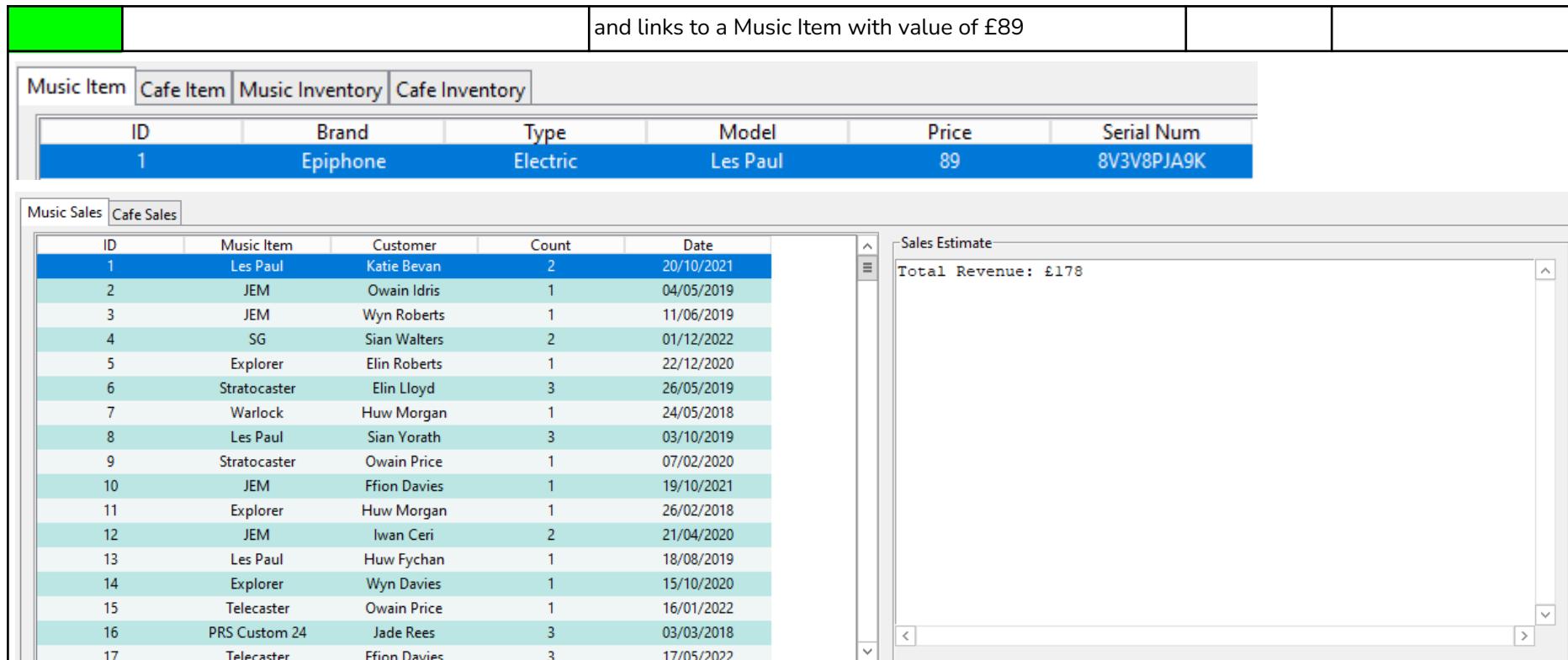
Sales record of 3 of that item correctly returns estimated revenue of £6.75				
				
				Sales Estimate Total Revenue: £6.75
2	Ensure system's Sales value estimate is correct for multiple records	Select one record in the Cafe Sales table where count = 3 and links to a Cafe Item with value of £2.25 And another record where count = 2 and links to a Cafe Item with value of £1.20		9.15
				

The Sales Estimate correctly returns £9.15

## Music Sales Estimate:

“Music Sales” tab tests to ensure calculations of the music sales income are correct:

Section: Sales		Tab: Music Sales		
Test Number	Purpose of Test	Data to Test	Expected outcome	Additional Notes
1	Ensure system's Sales value estimate is correct	Select one record in the Music Sales table where count = 2	178	

		and links to a Music Item with value of £89																																																																																																								
																																																																																																										
<table border="1"> <thead> <tr> <th>ID</th> <th>Brand</th> <th>Type</th> <th>Model</th> <th>Price</th> <th>Serial Num</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Epiphone</td> <td>Electric</td> <td>Les Paul</td> <td>89</td> <td>8V3V8PJA9K</td> </tr> </tbody> </table> <table border="1"> <thead> <tr> <th>ID</th> <th>Music Item</th> <th>Customer</th> <th>Count</th> <th>Date</th> </tr> </thead> <tbody> <tr><td>1</td><td>Les Paul</td><td>Katie Bevan</td><td>2</td><td>20/10/2021</td></tr> <tr><td>2</td><td>JEM</td><td>Owain Idris</td><td>1</td><td>04/05/2019</td></tr> <tr><td>3</td><td>JEM</td><td>Wyn Roberts</td><td>1</td><td>11/06/2019</td></tr> <tr><td>4</td><td>SG</td><td>Sian Walters</td><td>2</td><td>01/12/2022</td></tr> <tr><td>5</td><td>Explorer</td><td>Elin Roberts</td><td>1</td><td>22/12/2020</td></tr> <tr><td>6</td><td>Stratocaster</td><td>Elin Lloyd</td><td>3</td><td>26/05/2019</td></tr> <tr><td>7</td><td>Warlock</td><td>Huw Morgan</td><td>1</td><td>24/05/2018</td></tr> <tr><td>8</td><td>Les Paul</td><td>Sian Yorath</td><td>3</td><td>03/10/2019</td></tr> <tr><td>9</td><td>Stratocaster</td><td>Owain Price</td><td>1</td><td>07/02/2020</td></tr> <tr><td>10</td><td>JEM</td><td>Ffion Davies</td><td>1</td><td>19/10/2021</td></tr> <tr><td>11</td><td>Explorer</td><td>Huw Morgan</td><td>1</td><td>26/02/2018</td></tr> <tr><td>12</td><td>JEM</td><td>Iwan Ceri</td><td>2</td><td>21/04/2020</td></tr> <tr><td>13</td><td>Les Paul</td><td>Huw Fychan</td><td>1</td><td>18/08/2019</td></tr> <tr><td>14</td><td>Explorer</td><td>Wyn Davies</td><td>1</td><td>15/10/2020</td></tr> <tr><td>15</td><td>Telecaster</td><td>Owain Price</td><td>1</td><td>16/01/2022</td></tr> <tr><td>16</td><td>PRS Custom 24</td><td>Jade Rees</td><td>3</td><td>03/03/2018</td></tr> <tr><td>17</td><td>Telecaster</td><td>Ffion Davies</td><td>3</td><td>17/05/2022</td></tr> </tbody> </table> <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <b>Sales Estimate</b>        Total Revenue: £178     </div>					ID	Brand	Type	Model	Price	Serial Num	1	Epiphone	Electric	Les Paul	89	8V3V8PJA9K	ID	Music Item	Customer	Count	Date	1	Les Paul	Katie Bevan	2	20/10/2021	2	JEM	Owain Idris	1	04/05/2019	3	JEM	Wyn Roberts	1	11/06/2019	4	SG	Sian Walters	2	01/12/2022	5	Explorer	Elin Roberts	1	22/12/2020	6	Stratocaster	Elin Lloyd	3	26/05/2019	7	Warlock	Huw Morgan	1	24/05/2018	8	Les Paul	Sian Yorath	3	03/10/2019	9	Stratocaster	Owain Price	1	07/02/2020	10	JEM	Ffion Davies	1	19/10/2021	11	Explorer	Huw Morgan	1	26/02/2018	12	JEM	Iwan Ceri	2	21/04/2020	13	Les Paul	Huw Fychan	1	18/08/2019	14	Explorer	Wyn Davies	1	15/10/2020	15	Telecaster	Owain Price	1	16/01/2022	16	PRS Custom 24	Jade Rees	3	03/03/2018	17	Telecaster	Ffion Davies	3	17/05/2022
ID	Brand	Type	Model	Price	Serial Num																																																																																																					
1	Epiphone	Electric	Les Paul	89	8V3V8PJA9K																																																																																																					
ID	Music Item	Customer	Count	Date																																																																																																						
1	Les Paul	Katie Bevan	2	20/10/2021																																																																																																						
2	JEM	Owain Idris	1	04/05/2019																																																																																																						
3	JEM	Wyn Roberts	1	11/06/2019																																																																																																						
4	SG	Sian Walters	2	01/12/2022																																																																																																						
5	Explorer	Elin Roberts	1	22/12/2020																																																																																																						
6	Stratocaster	Elin Lloyd	3	26/05/2019																																																																																																						
7	Warlock	Huw Morgan	1	24/05/2018																																																																																																						
8	Les Paul	Sian Yorath	3	03/10/2019																																																																																																						
9	Stratocaster	Owain Price	1	07/02/2020																																																																																																						
10	JEM	Ffion Davies	1	19/10/2021																																																																																																						
11	Explorer	Huw Morgan	1	26/02/2018																																																																																																						
12	JEM	Iwan Ceri	2	21/04/2020																																																																																																						
13	Les Paul	Huw Fychan	1	18/08/2019																																																																																																						
14	Explorer	Wyn Davies	1	15/10/2020																																																																																																						
15	Telecaster	Owain Price	1	16/01/2022																																																																																																						
16	PRS Custom 24	Jade Rees	3	03/03/2018																																																																																																						
17	Telecaster	Ffion Davies	3	17/05/2022																																																																																																						

With the specified records set the Sales Estimate returns the expected value of £178

	Select one record in the Music Sales table where count = 2  and links to a Music Item with value of £89  And another record where count = 3  and links to a Music Item with value of £12		
2	Ensure system's Sales value estimate is correct for multiple records	214	

Music Item	Cafe Item	Music Inventory	Cafe Inventory		
ID	Brand	Type	Model	Price	Serial Num
1	Epiphone	Electric	Les Paul	89	8V3V8PJA9K
2	Schecter	Classical	JEM	12	7YDP6MMS7P

Music Sales	Cafe Sales			
ID	Music Item	Customer	Count	Date
1	Les Paul	Katie Bevan	2	20/10/2021
2	JEM	Owain Idris	3	04/05/2019
3	JEM	Wyn Roberts	1	11/06/2019
4	SG	Sian Walters	2	01/12/2022
5	Explorer	Elin Roberts	1	22/12/2020
6	Stratocaster	Elin Lloyd	3	26/05/2019
7	Warlock	Huw Morgan	1	24/05/2018
8	Les Paul	Sian Yorath	3	03/10/2019
9	Stratocaster	Owain Price	1	07/02/2020
10	JEM	Ffion Davies	1	19/10/2021
11	Explorer	Huw Morgan	1	26/02/2018
12	JEM	Iwan Ceri	2	21/04/2020
13	Les Paul	Huw Fychan	1	18/08/2019
14	Explorer	Wyn Davies	1	15/10/2020
15	Telecaster	Owain Price	1	16/01/2022
16	PRS Custom 24	Jade Rees	3	03/03/2018
17	Telecaster	Ffion Davies	3	17/05/2022

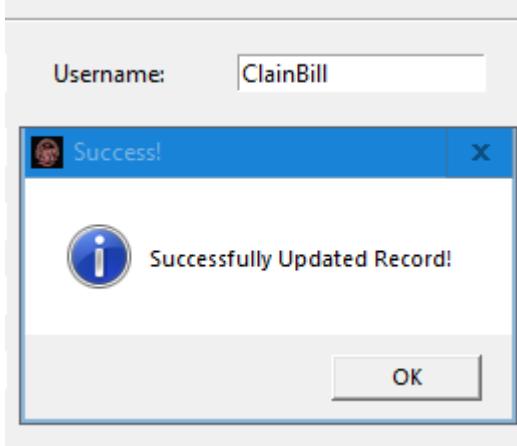
**Sales Estimate**

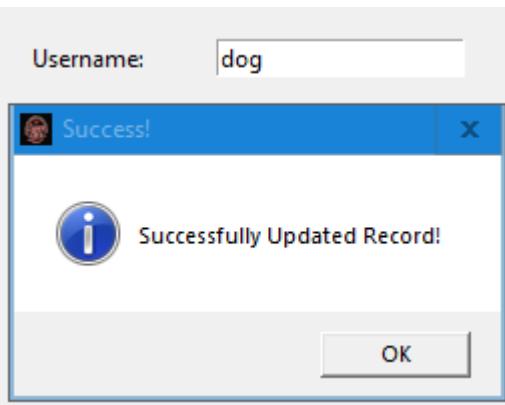
Total Revenue: £214

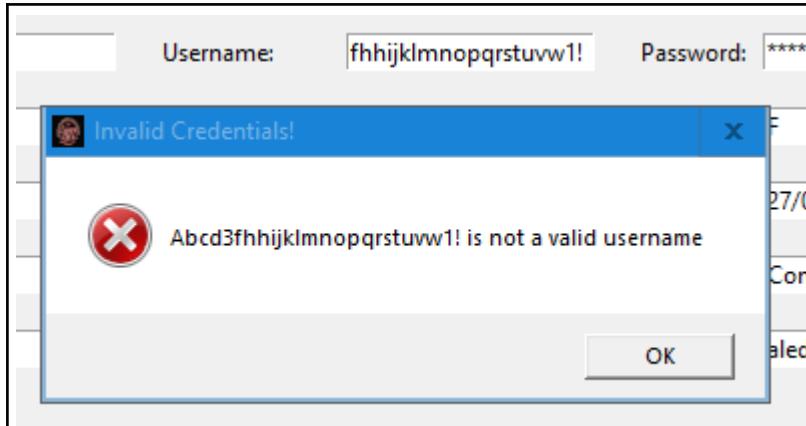
The Sales Estimate returns the expected value of £214

## VALIDATIONS

Any validation techniques that are reused will not need to be tested. This happens in several places such as for Town and County, or Forename and Surname and such.

Section: Records				Tab: Employee	
Test Num ber	Field to Test	Field Input	Test Purpose	Significance of Input Choice	Expected outcome
1	Username	ClainBill	Normal Data check Test should be okay	First Letter: Uppercase letter Type: only alphabetical letters Length: 9 satisfies $3 \geq \text{length} \geq 25$	Accepts Username
	 <p>A screenshot of a Windows application window. At the top, there is a text input field labeled "Username:" containing "ClainBill". Below it, a blue confirmation dialog box titled "Success!" displays the message "Successfully Updated Record!". An "OK" button is at the bottom of the dialog.</p>				
2	Username	dog	Lower Bound check	First Letter: Lowercase letter Type: only lowercase letters	Accepts Username

			Uses only one type of data and minimum length allowed	Length: 3 satisfies $3 \geq \text{length} \geq 25$	
					
3	Username	Abcd3fhhijklmnopqrstuvwxyz1!	Upper Bound check Uses all non-whitespace data types and is 25 characters long	First Letter: Uppercase letter Type: All non-whitespace data types used at least once Length: 25 satisfies $3 \geq \text{length} \geq 25$	Accepts Username



In this instance the system REJECTED the validation check when it should have ACCEPTED it.

```
def validateUsername(self):
    """Validates the entry as a username"""
    #Checks the entry contents with the regular expression:
    if not re.match(r"^[A-Za-z][\s]{2,23}$", self.get()):
        messagebox.showerror('Invalid Credentials!', str(self.get() + " is not a valid username"))
        self.valid = False
```

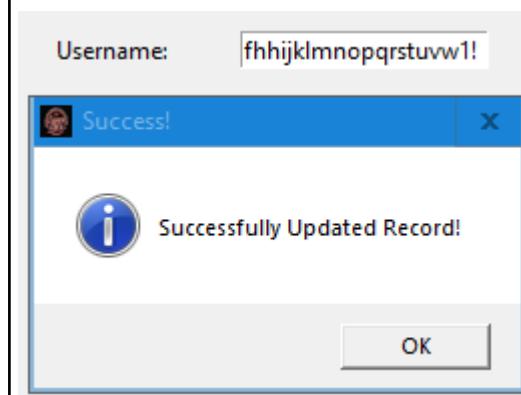
The above code was throwing the error.

I fixed it by changing the range in the curly brackets from 23 -> 24

This is the updated line:

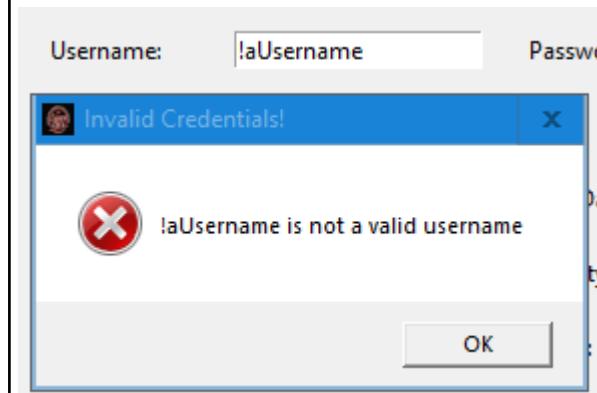
```
if not re.match(r"^[A-Za-z][\s]{2,24}$", self.get()):
```

Which now returns a successful validation attempt

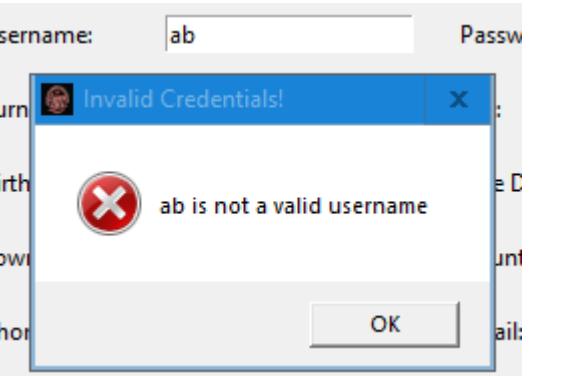


4	Username	!aUsername	Invalid Data check Should be invalid	First Letter: Symbol Type: N/A Length: N/A	Rejects Username for starting with symbol
---	----------	------------	---	--	---

The validation performs as expected and denies the validity of the username



5	Username	ab	Invalid Data check Should be invalid	First Letter: N/A Type: N/A	Rejects Username for being too short
---	----------	----	---	--------------------------------	--------------------------------------

				Length: 1 does not satisfy $3 \geq \text{length} \geq 25$	
					
6	Username	abcdefghijklmnopqrstuvwxyz	Invalid Data check Should be invalid	First Letter: N/A Type: N/A Length: 26 does not satisfy $3 \geq \text{length} \geq 25$	Rejects Username for being too long
7	Password	Passw0rd!	Normal Data check Test should be okay	Type: Contains at least 1 of every character type	Accepts Password

				Length: 9 satisfies $8 \geq \text{length} \geq 30$	
--	--	--	--	--	--

The system fails to validate the password field correctly



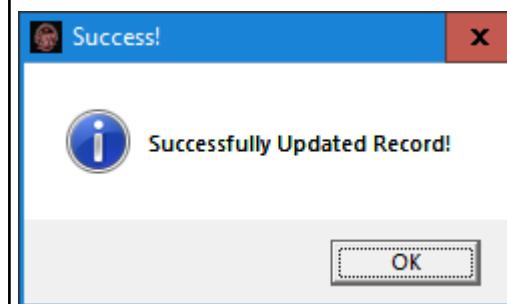
this password should be accepted as it follows the rules: length more than 8, at least 1 uppercase, lowercase, number, and special character

```
def validatePassword(self):
    """Validates the entry as a password"""
    #Password must be at least 8 characters long, have one uppercase letter, one lowercase letter, one number and
    one special character:
    if not re.match(r"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,30}$", self.get()):
        messagebox.showerror('Invalid Credentials!', str(self.get()) + " is not a valid password")
        self.valid = False
```

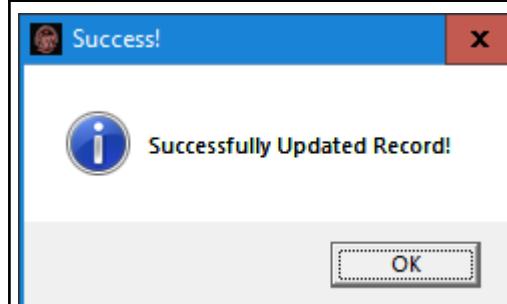
The regex code used was not working as intended. I decided to simplify the code as:

```
"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[^w\s])[^n]{8,30}$"
```

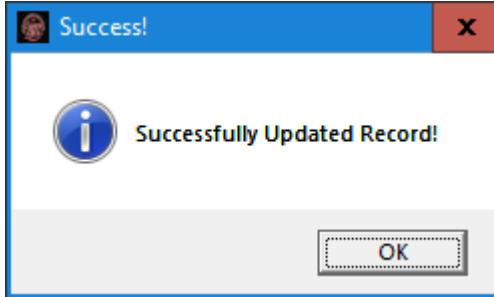
This now returns the expected outcome:



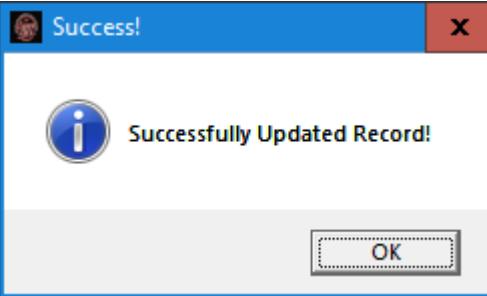
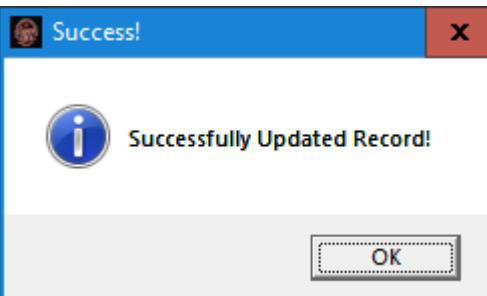
8	Password	Wrkpl3@s	Lower Bound check Uses only one of each data type and minimum length allowed	Type: Minimum number of types needed Length 8 satisfies $8 \geq \text{length} \geq 30$	Accepts Password
---	----------	----------	---	---	------------------

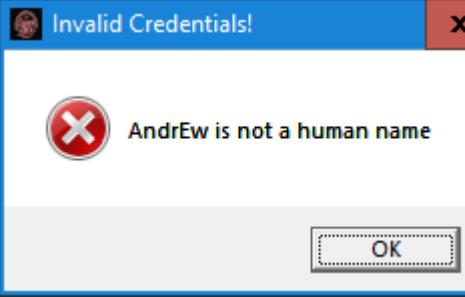


9	Password	Rc4!fragilisticexpialidocious	Upper Bound check Uses all non-whitespace data types and is 30 characters long	Type: All types needed Length 30 satisfies $8 \geq \text{length} \geq 30$	Accepts Password
---	----------	-------------------------------	---	--	------------------

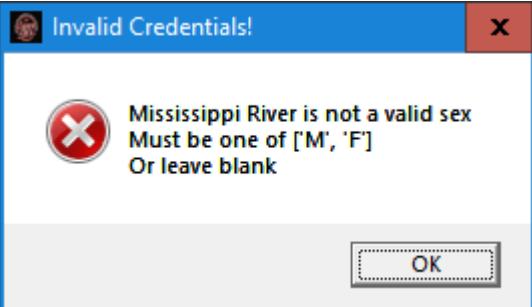
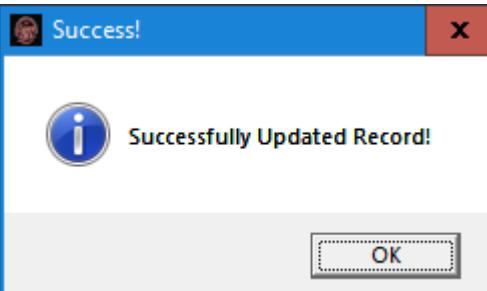
					
10	Password	Password	Invalid Data check Should be invalid	Type: Does not contain every data type needed	Rejects Password Needs at least 1 of all input types
					
11	Password	E^!1y	Invalid Data check Should be invalid	Length 5 does not satisfy $8 \geq \text{length} \geq 30$	Rejects Password Needs to be between 8 and 30 chars long

				
12	Forename	William	Normal Data check Test should be okay	First Letter: Uppercase letter Type: only alphabetical letters Length: 9 satisfies $3 \geq \text{length} \geq 25$ Accepts Forename
13	Forename	Bob	Lower Bound check Uses only one of each data type and minimum length allowed	First Letter: Uppercase letter Type: only alphabetical letters Length: 3 satisfies $3 \geq \text{length} \geq 25$ Accepts Forename

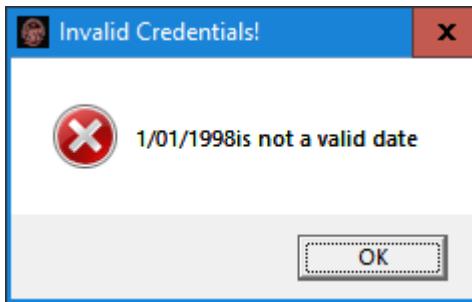
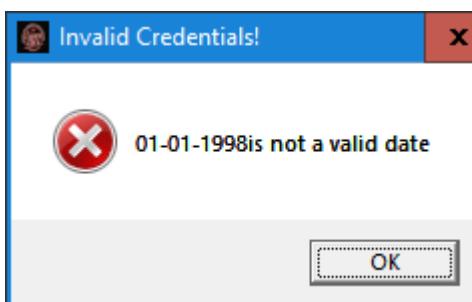
					
14	Forename	Christophermichaelsson	Upper Bound check Uses all non-whitespace data types and is 25 characters long	First Letter: Uppercase letter Type: only alphabetical letters Length: 9 satisfies $3 \geq \text{length} \geq 25$	Accepts Forename
					
15	Forename	timmy	Invalid Data check Should be invalid	First Letter: Lowercase letter Type: only lowercase letters Length: 5 satisfies $3 \geq \text{length} \geq 25$	Rejects Forename Must begin with Uppercase letter and proceed with only lowercase letters

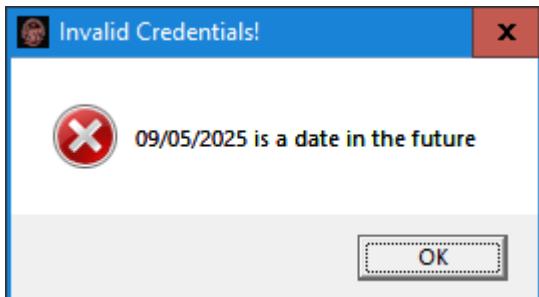
				
16	Forename	AndrEw	Invalid Data check Should be invalid	Type: Has uppercase letter after first letter  Rejects Forename Must begin with Uppercase letter and proceed with only lowercase letters
				
17	Forename	Ja	Invalid Data check Should be invalid	Length: 2 does not satisfy 3 >= length >= 25  Rejects Forename Must begin with Uppercase letter and proceed with only lowercase letters

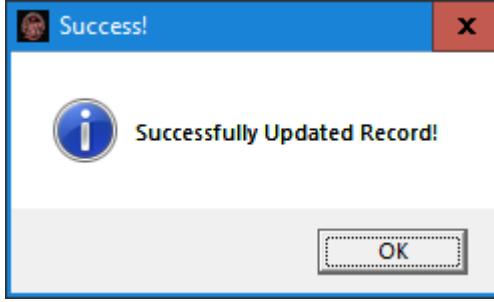
18	Sex	M	Normal Data check Test should be okay	M is in lookup check for sex	Accepts sex field
19	Sex	Mississippi River	Invalid Data check Should be invalid	"Mississippi River" is not in lookup check for sex	Rejects sex field Must be either 'M' or 'F'

				
20	Birthday	09/04/2005	Normal Data check Test should be okay	Type: Only numbers and '/' Format: Correctly formatted date Range: Is a date in the past Accepts Birthdate
				
21	Birthday	09/05/2023	Upper Bound check Is yesterday's date	Type: Only numbers and '/' Format: Correctly formatted date Range: Is the latest date still in the past Accepts Birthdate

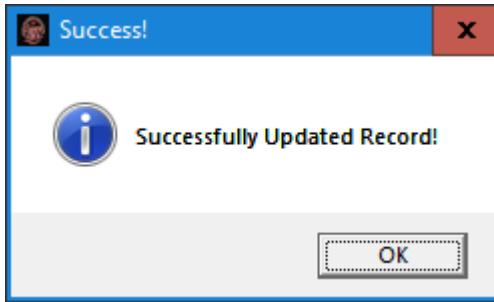
22	Birthdate	00/00/0000	Lower Bound check Is the earliest date representable in this format	Type: Only numbers and '/' Format: Correctly formatted date Range: Is earliest date possible	Accepts Birthdate
23	Birthdate	1/01/1998	Invalid Data check Should be invalid	Type: Only numbers and '/' Format: Incorrectly formatted date Range: Is a date in the past	Rejects Birthdate

				
24	Birthdate	01-01-1998	Invalid Data check Should be invalid	Type: Only numbers and '/' Format: Incorrectly formatted date Range: Is a date in the past Rejects Birthdate
				
25	Birthdate	first of january 1998	Invalid Data check Should be invalid	Type: Has alphabetical characters Format: Incorrectly formatted date Rejects Birthdate

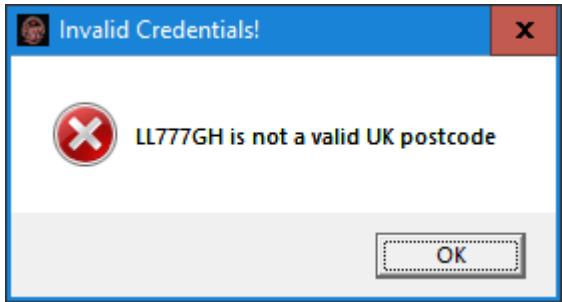
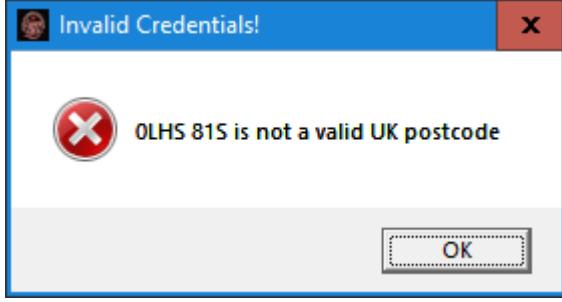
26	Birthday	09/05/2025	Invalid Data check Should be invalid	Type: Only numbers and '/' Format: Correctly formatted date Range: Is a date in the future Rejects Birthdate
				
27	Town	Llangefni	Normal Data check Test should be okay	Type: Only letters Format: Starts with capital letter Accepts Town

					
28	Town	Winchester-on-the-Severn	Normal Data check Test should be okay	Type: Only letters and hyphens Format: Starts with capital letter	Accepts Town
					
29	Town	Menai Bridge	Normal Data check Test should be okay	Type: Only letters and spaces Format: Starts with capital letters	Accepts Town

30	Town	Water 7	Invalid Data check Should be invalid	Type: Contains a number	Rejects Town
					
31	Town	tOkyO	Invalid Data check Should be invalid	Format: Does not start with capital letter	Rejects Town
					
32	Postcode	LL77 7GH	Normal Data check Test should be okay	Format: Follows correct postcode format of LL00 0LL	Accepts Postcode

					
33	Postcode	LL777GH	Invalid Data check Should be invalid	Format: Does not follow correct postcode format of LL00 0LL	Rejects Postcode
<p>This postcode should be rejected to be normalised with other postcodes</p> 					
<pre>def validatePostcode(self):     """Validates the entry as a UK postcode"""     if not re.match(r"^[A-Z]{1,2}\d[A-Z]\d{2}\$", self.get()):         messagebox.showerror('Invalid Credentials!', str(self.get()) + " is not a valid UK postcode")         self.valid = False</pre>					
<p>The regex code here is incorrect. The corrected regex code should be</p> <pre>"^LL77 \d[A-Z]{2}\$"</pre>					

With this the system behaves as expected:

				
34 Postcode	0LHS 81S	Invalid Data check Should be invalid	Format: Does not follow correct postcode format of LL00 OLL	Rejects Postcode
				
35 Phone Number	07726274201	Normal Data check Test should be okay	Type: Contains only numbers Format: Follows UK phone number format	Accepts Number Phone

 Success!					
 Successfully Updated Record!					
<input type="button" value="OK"/>					
36	Phone Number	07TwoSix4201	Invalid Data check Should be invalid	Type: Contains Letters Format: Does not follow UK phone number format	Rejects Phone Number
 Invalid Credentials!					
 07TwoSix4201 is not a valid UK phone number					
<input type="button" value="OK"/>					
37	Phone Number	620074160575	Invalid Data check Should be invalid	Format: Does not follow UK phone number format	Rejects Phone Number

	 Invalid Credentials!	X			
	 620074160575 is not a valid UK phone number				
			OK		
38	Email	william@freaks.org.uk	Normal Data check Test should be okay	Format: Follows conventional email format	Accepts Email
	 Success!	X			
	 Successfully Updated Record!				
			OK		
39	Email	william.freaks.org.uk	Invalid Data check Should be invalid	Format: Doesn't have @ symbol	Rejects Email
	 Invalid Credentials!	X			
	 william.freaks.org.uk is not a valid email address				
			OK		

40	Email	!william@freaks.org.uk	Invalid Data check Should be invalid	Type: Has non-@ symbol	Rejects Email
----	-------	------------------------	---	------------------------	---------------

This should be an invalid email address but the system accepts it.



```
def validateEmail(self):
    """Validates the entry as an email address"""
    #Checks the entry contents with the regular expression:
    #^ marks start of string
    #\S+ marks any number of non-whitespace characters
    #\$ marks the end of the string
    if not re.match(r"^\S+@\S+\.\S+$", self.get()):
        messagebox.showerror('Invalid Credentials!', str(self.get()) + " is not a valid email address")
        self.valid = False
```

The regex code here allows for any non whitespace character, when it should only allow for letters

The new regex code fixes this issue:

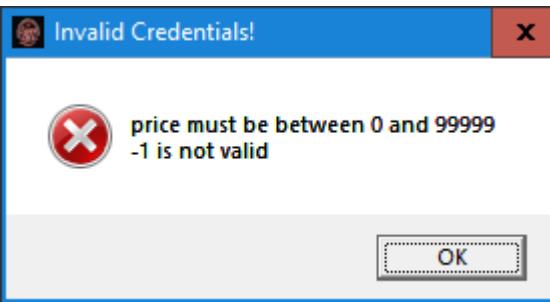
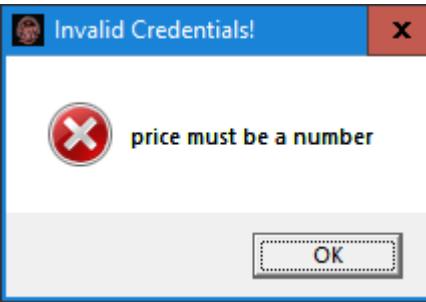
```
"^ [A-Za-z] +@[A-Za-z] +\.[A-Za-z] +$"
```

And the validation behaves as it should now:



Section: Stock			Tab: Cafe Item		
Test Number	Field to Test	Field Input	Test Purpose	Significance of Input Choice	Expected outcome
1	Price	12.5	Normal Data check Test should be okay	Type: Float Range: Within 0-99999	Accepts Price
Success!					
2	Price	0	Lower Bound check Uses minimum value allowed	Type: Float	Accepts Price

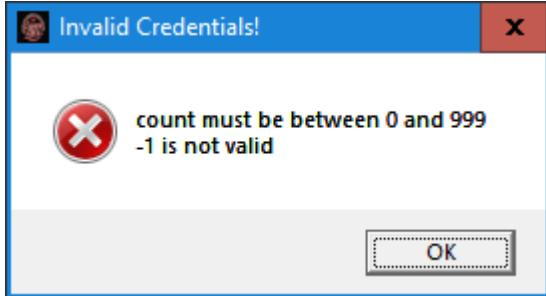
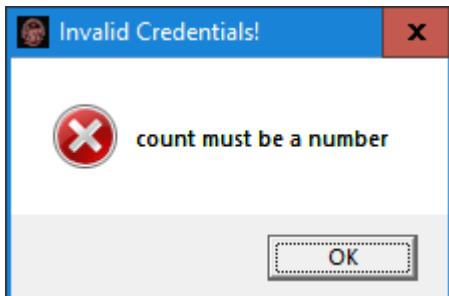
				Range: Within 0-99999	
3	Price	99999	Upper Bound check Uses maximum value allowed	Type: Float Range: Within 0-99999	Accepts Price
4	Price	-1	Invalid Data check Should be invalid	Type: Float Range: Not within 0-99999	Rejects Price Must be in range

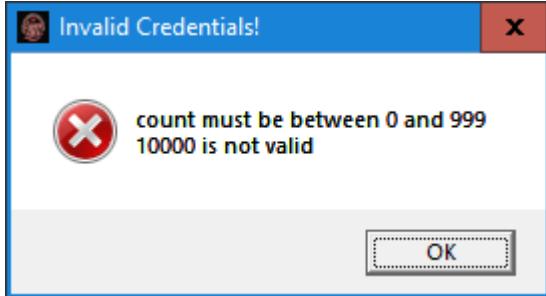
					
5	Price	twelve	Invalid Data check Should be invalid	Type: String	Rejects Price Must be a number
					
6	Price	100000	Invalid Data check Exceeds maximum value allowed	Range: Above 0-99999	Rejects Price Must be in range



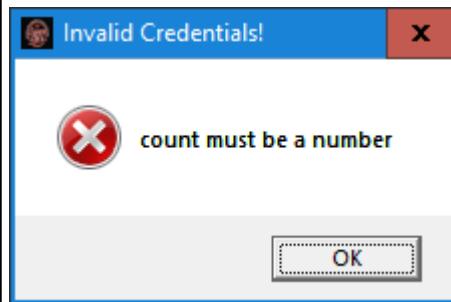
Section: Stock				Tab: Cafe Inventory	
Test Number	Field to Test	Field Input	Test Purpose	Significance of Input Choice	Expected outcome
1	Count	3	Normal Data check Test should be okay	Type: Integer Range: Within 1-9999	Accepts Price
Success!					
			Successfully Updated Record!		
2	Count	1	Lower Bound check Uses minimum value allowed	Type: Integer	Accepts Price

				Range: Within 1-9999	
3	Count	9999	Upper Bound check Uses maximum value allowed	Type: Integer Range: Within 1-9999	Accepts Price
4	Count	-1	Invalid Data check Should be invalid	Range: Not within 1-9999	Rejects Price Must be in range

					
5	Count	twelve	Invalid Data check Should be invalid	Type: String	Rejects Price Must be a number
					
6	Count	10000	Invalid Data check Exceeds maximum value allowed	Range: Above 1-9999	Rejects Price Must be in range

					
7	Count	3.5	Invalid Data check Should be invalid	Type: Not an integer	Rejects Price Must be in range
Should reject count, count can only be an integer					
					
<pre>if self.column.lower() == "count":     self.rangeCheck(min=1, max=999)</pre>					
The code performs a range check using float values. I had to change the function to call a rangeCheckint() method:					
<pre>if self.column.lower() == "count":     self.rangeCheckInt(min=1, max=999)</pre>					

With this the system performs as intended:



## OPERATIONS TESTING

The various operations the system can provide need to be tested to ensure they work.

Evidence for these tests will be given in the PowerPoint with references to the test number

The following tests ensure the login system functions correctly

Window: Login				
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes
1	Ensure system will restrict incorrect credentials	Enter "admin" in the username field. Enter "Password" in the password field. Click the "Login" button or press Enter	A popup will tell the user their credentials were incorrect	
See PowerPoint "Login test 1" for evidence				

2	Ensure system will allow correct credentials	Enter "admin" in the username field. Enter "admin" in the password field. Click the "Login" button or press Enter	The system allows the user and gives them access to the "Manager Dashboard" as noted in the title of the window	
See PowerPoint "Login test 2" for evidence				
3	Ensure the system will provide employees with a different level access to managers	Enter "employee" in the username field. Enter "Passw0rd!" in the password field. Click the "Login" button or press Enter	The system allows the user and gives them access to the "Employee Dashboard" as noted in the title of the window	"employee" is an example employee record I have preloaded in the system with these credentials
See PowerPoint "Login test 3" for evidence				
4	Ensure the system will provide teachers with a different level access to managers	Enter "teacher" in the username field. Enter "Passw0rd!" in the password field. Click the "Login" button or press Enter	The system allows the user and gives them access to the "Teacher Dashboard" as noted in the title of the window	"teacher" is an example employee record I have preloaded in the system with these credentials
See PowerPoint "Login test 4" for evidence				
5	Ensure the system will provide students with a different level access to managers	Enter "student" in the username field. Enter "Passw0rd!" in the password field. Click the "Login" button or press Enter	The system allows the user and gives them access to the "Student Dashboard" as noted in the title of the window	"student" is an example employee record I have preloaded in the system with these credentials

See PowerPoint "Login test 5" for evidence
--

These records work identically in all tabs. The only difference is different users might not have access to all commands in every tab

Section: Records		Tab: Employees			
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes	
1	Ensure Clear Entries Button Works	Click on "Clear Fields" button when all fields are filled with data	All entries in this record are cleared		
See PowerPoint "Records Test 1" for evidence					
2	Ensure user can Add Records	Fill in all entries and click "Add Record" button	A popup confirms the record was added and the new record appears in the treeview grid		
See PowerPoint "Records Test 2" for evidence					
3	Ensure user can Edit Records	Click on a record and change any entry other than the ID Click on the "Update" button	A popup confirms the record was updated and the updated record appears in the treeview grid		
See PowerPoint "Records Test 3" for evidence					

			A popup asks the user to confirm the user wants to delete the record. Then when confirmed the record is removed from the treeview and the entries are cleared		
4	Ensure user can Delete Records	Click on a record and click on the "Delete" button. Click "OK" on the popup that will appear			
See PowerPoint "Records Test 4" for evidence					
5	Ensure user can Search Records with one search parameter	Type a value into any record field that would make sense to appear (such as 'M' in the 'Sex' field) Click on the "Search" button	The treeview should update to only show results that have that exact data in the specified column		
See PowerPoint "Records Test 5" for evidence					
6	Ensure user can Search Records with multiple search parameters	Type another value into any other record field that would make sense to appear (such as 'Llangefni' in the 'Town' field) Click on the "Search" button	The treeview should update to only show results that have that exact data in all the specified columns	If no results appear adjust the filters to be more common	
See PowerPoint "Records Test 6" for evidence					
7	Ensure the user can perform "After" date range filters	With no entries filled (push the "Clear Fields" button) put a date in the "After Date" field of the birth date range check in the "Range Check" frame on the right.	The treeview should filter out any results that had a birthdate before the date	If no results appear try adjusting the range	

		Click on "Filter"	entered		
	See PowerPoint "Records Test 7" for evidence				
8	Ensure the user can perform "Before" and "After" date range filters	With no entries filled (push the "Clear Fields" button) put a date in the "After Date" and "Before Date" field of the birth date range check in the "Range Check" frame on the right. Click on "Filter"	The treeview should filter out any results that had a birthdate outside the range given		
	See PowerPoint "Records Test 8" for evidence				
9	Ensure the user can perform date range filters to their current search	With the same filter applied, apply a "Search" filter by typing a sensible value into a record field. Click on "Filter" inside the "Range Check" frame on the right with the filled in range	The treeview should display only results that match all of the filters applied	If no results appear try adjusting the search and filter parameters to be more broad	
	See PowerPoint "Records Test 9" for evidence				

Toolbar Menu at top of window				
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes
1	Ensure Autosave works	<p>Make a change to a record (such as updating, adding or deleting a record)</p> <p>Hold down the "Alt" key on the keyboard and then simultaneously press the "F4" key to force close the program.</p> <p>Reopen the program and the changes should be saved.</p>	When the program is opened again the changes made before force closing the program should be maintained.	
See PowerPoint "Toolbar Test 1" for evidence				
2	Ensure "Logout" command works	Click on "File" and then hover over and select "Logout" from the drop down menu.	The program should save and close the main program window, and the login window should automatically open	
See PowerPoint "Toolbar Test 2" for evidence				
3	Ensure "Exit" command works	Click on "File" and then hover over and select "Exit" from the drop down menu.	The program should save and close the main program window	
See PowerPoint "Toolbar Test 3" for evidence				

4	Ensure "Import" command works	Click on "Edit" and then hover over and select "Import" from the drop down menu.	A new import window should appear above the primary application allowing.	The "Edit" dropdown menu is only accessible by the Managers and Employees. So MAKE SURE YOU'RE LOGGED IN AS SUCH
See PowerPoint "Toolbar Test 4" for evidence				
5	Ensure "Export" command works	Click on "Edit" and then hover over and select "Export" from the drop down menu.	A new export window should appear above the primary application allowing.	The "Edit" dropdown menu is only accessible by the Managers and Employees. So MAKE SURE YOU'RE LOGGED IN AS SUCH
See PowerPoint "Toolbar Test 5" for evidence				

Import Window				
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes
1	Ensure importing works correctly	Create a '.csv' file that contains the text in the Additional Notes of this test. Select the "Employee" table in the selection box. Click the "Import" button and select the file you created in the file selection window that appears.	A popup says success and tells the user to refresh the table to see results. If the test was successful the imported record should be seen at the end of the "Employees" table when the "Search" button is pressed with no entries filled	username,password,forename,surname,birthdate,job_description newUser,newPassw0rd!,Timothy,Tester,01/01/1980,employee
See PowerPoint "Import Test 1" for evidence				
2	Ensure importing handles errors correctly	Create a '.csv' file that contains the text in the Additional Notes of this test. Select the "Employee" table in the selection box. Click the "Import" button and select the file you created in the file selection window that appears.	A popup says the file was invalid	username, password newUser,newPassw0rd!,Timothy,Tester,01/01/1980,employee
See PowerPoint "Import Test 2" for evidence				

Export Window				
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes

1	Ensure exporting works correctly	Select any table in the selection box. Click the "Export" button and select a file location	A file is created in the location specified	
See PowerPoint "Export Test 1" for evidence				

Section: Lessons		Tab: Student Bookings		
Test Number	Purpose of Test	Instruction	Expected outcome	Additional Test Notes
1	Ensure calculations are successfully exported	Select either one or multiple records in the treeview by shift-clicking.  Click on the "Calculate" button in the "Export" frame on the right.  Then click on "Export"	A "LessonEstimate.txt" file is created in the directory of the application	
See PowerPoint "Student Bookings Test 1" for evidence				
2	Ensure calculations are successfully added to the existing file	Again, Select either one or multiple records in the treeview by shift-clicking.  Click on the "Calculate" button in the "Export" frame on the right.  Then click on "Export"	The "LessonEstimate.txt" file is appended to with the new calculation made	
See PowerPoint "Student Bookings Test 2" for evidence				

# Evaluation

## REVISED SPECIFICATION

### **1. User Access and Login System:**

- 1.1 - Restrict app to allow different levels of access for managers, employees, teachers, and students.
- 1.2 - Login system to authenticate users and grant appropriate permissions. Users will only have access to the features of the GUI their access levels allow.

### **2. Customer Records:**

- 2.1 - Managers can create, update, and delete validated customer records with their personal information.
- 2.2 - Display list of customers with personal information to managers in a searchable and sorted table.
- 2.3 - Display the purchase history of customers in a searchable and sorted table.
- 2.4 - Filter records from a range of birthdates

### **3. Employee Records:**

- 3.1 - Managers can create, update, and delete validated employee records with their personal information.
- 3.2 - Display list of employees with personal information to managers in a searchable and sorted table.
- 3.3 - Employees can edit their own records.
- 3.4 - Filter records from a range of birthdates and hire dates

### **4. Teacher and Student Records:**

- 4.1 - Managers can create, update, and delete validated teacher and student profiles with their personal information.
- 4.2 - Display list of teachers and students with personal information to managers in a searchable and sorted table.
- 4.3 - Teachers and students can edit their own records
- 4.4 - Teachers and managers can view the progress and attendance of students in their respective lessons.
- 4.5 - Filter records from a range of birthdates and hire dates (for teachers)

**5. Sales and Inventory Management:**

- 5.1 - Managers can create, update, and delete validated stock items for both the music store and the cafe with detailed information, such as product name, description, and price.
- 5.2 - Record inventory levels over a period of time.
- 5.3 - Track sales from user input.
- 5.4 - Automatically update inventory levels with sales.
- 5.5 - Allow managers to add items to inventory levels.
- 5.6 - Filter records from a range of date options and price options

**6. Student Lessons:**

- 6.1 - Managers can create, update, and delete each student's lesson bookings.
- 6.2 - Managers and teachers can view searchable and sorted lists of student reports.
- 6.3 - Teachers can submit lesson reports after each lesson.
- 6.4 - Managers can create lesson plans.
- 6.5 - Managers and teachers can view a searchable and sorted list of lesson plans.
- 6.6 - Filter records from a range of lesson dates and costs

**7. Reporting and Data Analysis:**

- 7.1 - Calculate and display sales report given a date range.
- 7.2 - Calculate and display value of inventory.
- 7.3 - Display estimate of revenue from student lessons.
- 7.4 - Support exporting calculations to .txt or .csv format.

**8. Data Security and Integrity**

- 8.1 - Encrypt all stored data to prevent personal information being leaked.
- 8.2 - Make routine backups of all stored data.
- 8.3 - Validate all user input.

# Introduction

Overall the project has taken a large amount of time and effort to complete, but in doing so I have learned crucial information about the process of developing an application for a company wishing to upgrade its system to a digital solution.

I believe that my choice of program and environment was well decided and that I have succeeded in producing a highly functioning program with quality documentation and a good design.

Liz Pryde was a helpful hand in all aspects of the project. She ensured that my program remained on track with the company's needs and she also reassured me that the program was developing with good quality when I was unsure of its progression.

If I was to approach the project again I would use different methodologies and strategies now that I have learned from experience in developing a functioning application. I believe that the experience I have gained from this task will lead me to have a better understanding of working in a software development workflow and also managing my own skills as an individual in the process of completing a long and complicated task.

# The Programming Language

## PYTHON

Python was chosen because it is a simple programming language that has very powerful functionality. It supports Object Oriented Programming which opens the door to more powerful solutions to problems and can also make code much easier to navigate.

Additionally, Python comes with many pre-installed standard modules that expand its use case drastically. During the development of my project I made use of a massive amount of modules, some of them had specific use cases that were only needed for a single task, but having them available was crucial to the vast amount of features my program had to offer.

The most obvious use case for python's modules was the Tkinter library which was used for every Graphical User Interface the user could interact with. Tkinter itself is not very complex but its simplicity allows for scalable applications to be built on its well designed features, much like Python itself.

Something else that Python offers that has provided extremely useful in many cases is the Try/Except feature that allows the user to define their own error handling routines. I wish I had made more use of this syntax in my program as I had only just discovered its use case near the end of my development, but it did still prove to make some aspects of the system simpler to handle with the many issues that could arise.

Another reason Python was chosen over other high level programming languages is that Python is very easy to learn. I had nearly no experience with programming prior to this project, however now I feel comfortable with using Python's many commands and I have gained a better understanding of programming languages in general as a result of my deep usage of Python.

Python's programming principles can be carried over to many other languages and have proved to improve my problem solving skills. Upon beginning programming I would be stuck with how to solve simple problems but now I can solve issues by breaking down the problem via abstraction and terminating the cause to find a solution much quicker than if I had to turn to an external source

# INTEGRATED DEVELOPMENT ENVIRONMENT

At school I was using Python's own IDLE editor to write and execute python code in. This Integrated Development Environment (IDE) did offer many crucial features to me as I developed the application, but was lacking some features that more modern IDEs offer today.

At my home when I was working on the project I made frequent use of the advanced text editor Visual Studio Code. VS Code is not an IDE but does offer many of the features a developer requires from an IDE and also offers some extra features.

There were however some issues with using VS Code to develop such an old version of Python with. Python 3.4.2 falls behind the legacy support of VS Code's debugging features. This meant that when I needed to execute any python script I had to either switch the environment to using Python's IDLE IDE, or I would simply run the code with no debugging tools with VS Code.

This was however balanced out by the fact that VS Code offers many tools that sped up the development process:

Tool	Tool Description	Tool Use
Automatic Bracket Closing	VS Code automatically closes brackets, square brackets, curly brackets, and quotation marks around parameters.	When typing a function name or casting a variable VS Code allowed me to highlight the code and it would automatically place my enclosing brackets around the selection. This speeds up the development process by reducing the number of tedious processes.
Multiple Cursors	In VS Code by holding Alt and clicking you can place multiple cursors.	Having access to multiple cursors simultaneously makes editing multiple instances of the same variable or method much quicker as the process of

		editing that object only has to be done once.
Find/Replace	VS code has advanced integration of the commonly used tool allowing users to find and replace text across the entire document or in a selection.	This tool allows me to replace every instance of a group of text or variable name across the entire document. This speeds up the process of individually changing each instance of text, and also ensures that I have changed every single instance of that text and that I haven't missed one.
Jump to Line	By pressing a hotkey the program can jump instantly to any line of code.	This feature paired with the debugging tools allows me to find and fix issues in the large codebase very quickly. I can jump directly to the line that the console has warned an error has occurred in and see what the issue was.
Breakpoints	VS code can place various breakpoints across the entire document. Breakpoints will pause execution allowing me to see values in variables.	Breakpoints simplify the process of debugging as it allows me to stop the program from reaching a line where I know an error occurs so I can inspect the state of variables before the error occurs and find the cause.

VS Code also makes much more rich use of syntax highlighting. The program offers a diverse set of themes to choose from to match my specific style. While arguably only an aesthetic choice having good colour-coding has led to better efficiency when navigating the codebase.

Another feature of VS code is its integration with version control software. I decided to use GitHub for this project because it allowed me to backup each version of the code onto an external cloud based server. The server hosted backup meant that I could keep up to date with my code whether I was working from home or at school.

The version control allowed me to look back at previous versions of the software and examine what had caused my code to stop working when I had seemingly not changed the related classes or methods.

Overall using VS Code as my primary work environment proved to be very beneficial to the production of the code. Additionally, I was not limited to the drawbacks of using VS Code as I also had IDLE available to me whenever I needed to run debugging tools on my program.

# Self Evaluation of Solution

I feel like the final solution I was able to produce was a good testament to how I am able to break a problem down into sub objectives and complete a broad goal.

That being said. I am more aware than anyone of my own shortcomings during the planning and execution of the project.

My overall impression is that I have completed the task to a high standard, but could have had better preparation and a higher standard of keeping to deadlines that would have ensured that I would have completed every objective in the specification.

Whilst completing most of the objectives there were a few that I was not able to complete in time for the deadline:

## **5.4 - Automatically update inventory levels with sales**

I am disappointed that I was not able to complete this objective in time for the deadline. This would have been a helpful feature for Lizzy Lee's Rock Cafe to make use of, but I was not able to make a clear enough plan of how to integrate this feature, and as a result it never made the final cut, or was even placed on a prototype.

A system that takes minimal input to keep track of real inventory values could have saved a lot of effort on behalf of the staff of Lizzy Lee's. This will need to be taken as a priority update for future versions of the software.

## **7.4 - Support exporting calculations to .txt or .csv format**

I was able to implement this objective, however Liz Pryde made a suggestion after using the prototype that I should expand this objective to the Sales and Inventory calculations that are made automatically when the user selects records.

However I do feel that this feature would not have been that beneficial to include compared to the other features that I was able to implement in the digital design in its place.

The final product itself I feel is lacking a little bit of design. Whilst it does satisfy my 'industrial itch' I admit that the design is not the most appealing for daily use by a company that runs a rock cafe.

If I had a chance to do the project over I would have spent more time with the planning stage rather than the development so that I could fully flush out what exactly the stakeholders needed in every aspect of the system and not just its features.

That being said, I am very happy with the number of features I was able to cram into this project.

The final design had 13 tables to handle and was able to link together many of them seamlessly. The login system looks at both the actual table the credentials matched to, and if the table was EMPLOYEE it would perform an SQL query to retrieve the user's company role. Depending on all of this the system would then seamlessly grant the user access to whatever features they are in need of.

# My Performance

Throughout the program I was able to overcome a number of obstacles that I had never faced before in a professional setting. The project has pushed me and forced me to come up with creative solutions to problems in order to keep to deadlines and balance my other responsibilities.

First of all I have never had to deal with a client in such a manner until undertaking this project. Performing professional interviews and delivering questionnaires to staff that I would be working around was an experience that I made an attempt to match in a professional setting.

When interacting with the stakeholders I made sure that there was clear communication between us, even if it was through indirect digital means. I believe that this has led to a clear understanding of what the program had to entail and the ability to receive good feedback for my work.

However it is true that I was not confident when conducting research into similar solutions.. It was difficult for me to find examples of similar solutions. I feel like the preparation stage was one of my weakest areas of project development.

At the beginning of the project I was not fully aware of what I was expected to do for each stage of development which held me back significantly for some time. When I was able to understand what exactly I was required to do I took time to ensure that my work was of a high quality, and specifically ensured that the design and flow of my documentation was easy to understand and led to what I hope to be a pleasant experience.

Some major issues I overcame during this project was:

## ENCRYPTION

I had planned for the program to make use of asymmetric encryption as I knew this was a very secure form of encryption. However, during the implementation phase I quickly discovered that this was a mistake and that it was unnecessary to make use of such a powerful algorithm when it was unnecessary to do so.

Symmetric encryption, I discovered, is just as secure as asymmetric encryption, except it only uses one key. I had naturally assumed that one key meant that it was less secure, but in fact it only makes a difference if the system is transmitting data over a network which my system had no need to do - therefore I would be able to switch over to symmetric encryption without any fear of losing too much security in favour of a much faster program.

The only issue I had with implementing this was that I had to scrap the effort I put into learning asymmetric encryption which included me discovering and utilising recursive algorithms and powerful mathematics - however it was necessary for the stakeholders of Lizzy lee's Rock cafe.

## SECURE STORAGE OF FILES

I had to make a decision of how I would store my non-volatile data in a method that allowed for the program to hide the sensitive data from prying eyes without losing any performance.

My initial idea was to store an encrypted .db file and decrypt the data as it is needed from the file, however I came to the conclusion that this was not a quick solution and it also had issues that could jeopardise the security of the files it was storing such as being susceptible to brute force attacks that could attempt to decrypt obvious fields such as IDs or a column that contained only 'M' for Male and 'F' for Female if opted for using a caesar shift encryption method.

Overall I decided that this was not a good solution to the issue of storing secure files. In the end I realised that sqlite had the ability to dump its data into a text friendly format using

“iterdump”. I realised that instead of storing a ‘.db’ file I could simply handle the sqlite data inside computer memory using :memory: and then dumping the schema and data into an encrypted text file.

When the data is needed again the program would know how to decrypt the file, and then it could parse the data back into a usable SQL schema now stored in memory. This way sensitive data is never left unsecured on non-volatile hardware which would reduce the risk of a data breach.

## SQL QUERIES

Having never used SQL before I have gained a very good understanding of the syntax and commands the language uses to retrieve data from complex structures.

One of the most challenging aspects of the program was finding a way to merge the results of multiple SQL commands to satisfy complicated data filtering. This is most evident when a teacher is logged in and needs to perform a birth date range filter on their students who live in Llangefn and have the title Mr.

This example requires the system to match a superseding filter that filters out student records to only display students who have the teacher logged in as their own teacher in a lesson booking, with the date-range filter, with any search filters the user has applied in the entry fields.

Managing these three filters was difficult but by drawing out a diagram connecting the filters in order of importance and by researching SQLite query commands I was able to come up with a solution that works excellently.

# OBJECT ORIENTED PROGRAMMING

This is an area I feel conflicted about. Originally I was under the impression that I had handled the challenge of utilising Object Oriented Programming very successfully, and that in doing so I had reduced the number of lines in my code significantly by keeping it DRY.

However this changed when I had to keep on adding features that led to the program quickly increasing in size rapidly until it almost had reached 4 times its size. Granted, I was still a ways away from completing all the objectives in the specification but what I thought was the entire skeleton of the program had ended up growing into a frankenstein that exceeded a size that I had hoped for.

Again, I put down this disappointment to a lack of preparation beforehand. The entire program makes half good use of OOP and for the other half I have had to repeat many lines of code to get features implemented without error.

Looking back at what I have done I have learned much about the good use of OOP and also the dangers of a poor OOP implementation.

# Similar Available Systems

From scouring the internet I have found a couple of similar systems to the application I have developed:

## Mindbody

Mindbody is a management software for businesses specialised in the wellness industry. They offer scheduling, appointment booking, payment processing, and other features to their clients. Overall, their system is much more polished and has a higher quality interface compared to my solution but their service does not offer the exact features that my application provides that Lizzy Lee's requires.

Mindbody is a much more generalised product for companies who offer a wellness program to their employees and desire to manage how their employees interact with the wellness program.

## Teachworks

Teachworks is a software designed for music tutors to help them manage their students. It is designed for individuals who manage their own work and need to keep track of their own students' progression and their own bookings.

Teachworks does not offer the specific solution to Lizzy Lee's problem that I have been able to solve with my application - however Teachworks is a much higher quality product that has been tested on a large number of devices and is deployed across many operating systems and languages.

My own solution is not as polished as either of these similar systems, but it is able to provide an exact solution to the problem that the Lizzy Lee's Rock Cafe company is facing.

# Change Of Approach

If I was to do this project again I would allocate more of my time to preparation and planning, and more crucially I would **stick to my deadlines**. As I have worked on this project I have been fighting against the clock for various reasons whether it is other subjects in school I am focusing on, or extracurricular activities that require my attention , or family affairs that never cease to bombard us.

I believe that a good timetable to work with and the discipline to stick to it could have saved some poor performance from my input and helped me to maintain the standards that I aim to keep.

I would also like to consider adapting the solution to a more readily available interface; a website to host the company's management software could have been beneficial to the company as they would be able to access and make changes to the system's resources remotely. Furthermore, it would also bring the interface to more people as a website can be accessed from any device, whereas the current system can only run on computer operating systems that can run Python with Tkinter.

One feature that I didn't have the opportunity to implement was a file archive system. Archiving files would have allowed the system to run more smoothly as its database grows because currently all data has to be stored in memory for the program to run.

Utilising an archiving system would free up system memory for the rest of the program and other programs to run more smoothly, and archived files would also stop the system from accessing infrequently used files thus slowing down the access times.

Finally, if I was to approach this project again, I would write out a detailed plan of every class my system would need so that I would not run into any issues of integrating nested classes for the most efficient code. How I approached classes should have been given more care and I have learned to plan better to avoid this in the future.