

Contents

User Manual	2
1. Introduction	2
2. Installation.....	2
2.1. The System	2
2.2. Prerequisite Installation.....	2
2.3. System Installation	3
2.4. Running the System.....	4
3. Tutorial.....	5
3.1. Introduction	5
3.2. Assumptions.....	5
3.3. Tutorial Questions	6
3.4. Saving	6
4. Errors	17
4.1. User error	17
4.2. Program bugs.....	19
4.3. Incomplete sections of code.....	19
5. System Recovery.....	20
5.1. Backing up data	20
5.2. Restoring data	22

User Manual

1. Introduction

In this manual I will go through how to properly install, run, use, trouble shoot and backup my application.

The user will require a computer to the correct specifications, discussed in section 2.1, a CD-ROM containing the application installer and basic understanding of the operating system Windows 7.

2. Installation

2.1. The System

In my design I specified that my application will primarily run on computers running Windows 7, but the application was designed for the computer specifications below.

- Intel Core i5 running at 3.3 GHz
- 4GB DDR3 running at 2000 MHz
- SATA III 600 GB Internal HDD
- CD-ROM Drive
- Motherboard onboard graphics

The operating system of this computer will also need to be Windows Ultimate 64-bit, and doesn't require any internet connection to install or run.

2.2. Prerequisite Installation

My application does not require any programs installed on the computer apart from the application itself. If the application's source code were to be distributed, instead of an executable version, then the user would need to install the following.

- Python 3.2 Windows 64-bit
- PyQt (for Python 3.2 Windows 64-bit)

My program was never intended to be run on any Mac or Linux based computer, because I never specifically implemented any cross-platform functionality but theoretically PyQt and Python can be installed on both Mac and Linux computers. So it is possible.

2.3. System Installation

To install my application, the user will first need to be given a medium that can store a minimum of 6.80 MBs of data. This medium could be a CD-ROM, DVD, USB, or if the system has internet connection, via the internet. But for the following example I will assume the user acquired the application through the preferred distribution method, CD-ROM.

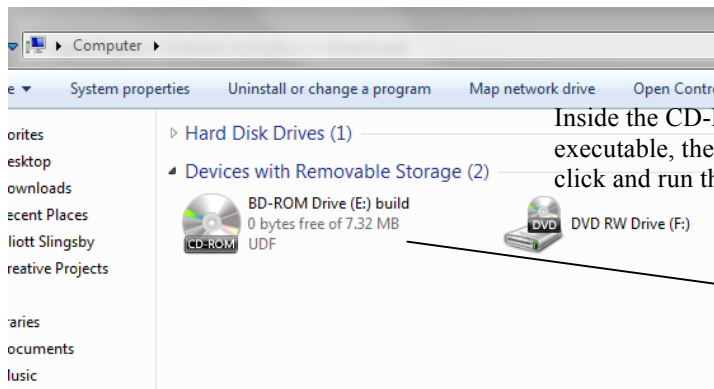
After inserting the CD-ROM, the user will need to navigate to my computer, and open the contents of the CD-ROM.

This is the path the installer will extract the files to.

This is the CD-ROM with the files needed to run on. The user just needs to double click on the icon, opening the contents.

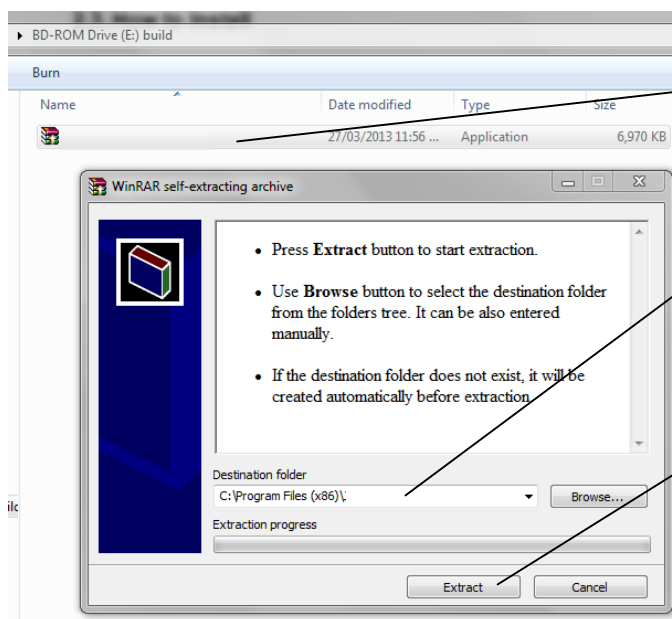
If everything seems okay, the extract

button will commit the extraction and install the program.



Inside the CD-ROM is the installer executable, the user just need to double click and run the installer.

Inside the CD-ROM is an executable file named "jeweller_installer.exe", upon running it, the user will be displayed the extract window.



The installer is a WinRAR self extractor archive, which can be ran on any system even if it doesn't have WinRAR installed. Once the installer has finished, the extractor will close automatically.

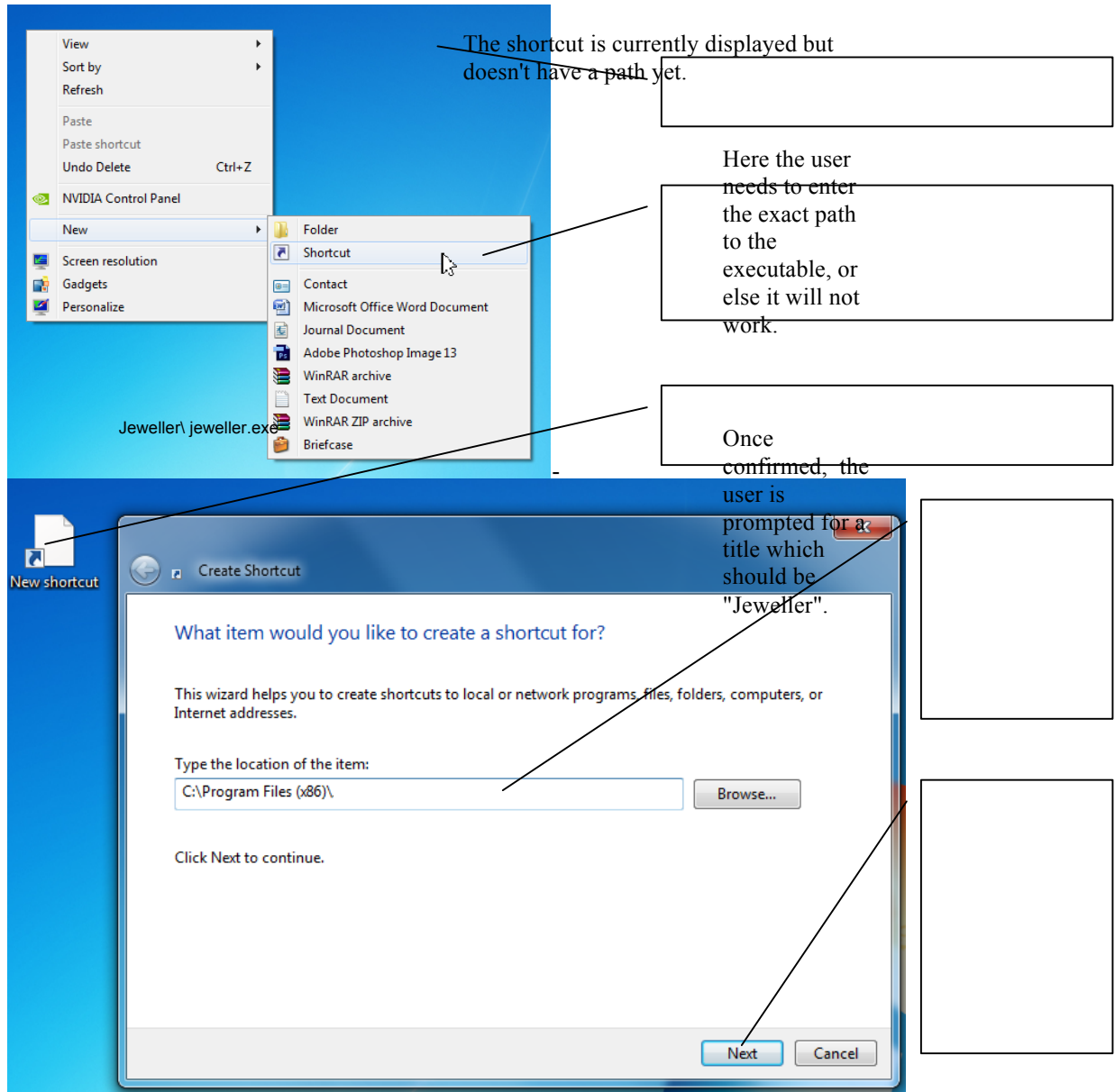
2.4. Running the System

Jeweller

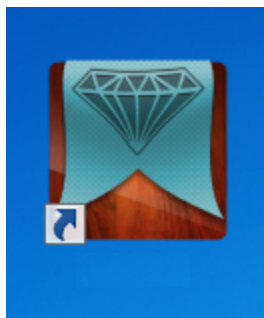
To run the program, the user could either navigate to "C:\Program Files (x86)\Jeweller", and run "jeweller.exe", or for ease of use over time, the user could create a shortcut on the desktop to "C:\Program Files (x86)\Jeweller\jeweller.exe", and title it "Jeweller". This is suggested.

The user needs to right click anywhere on the desktop, go to new, and then shortcut.

The create shortcut window should then open.



When finished, the icon should then default to the icon of the executable, which is confirmation that it installed correctly.



3. Tutorial

3.1. Introduction

In this section I will go through how to completely utilize every function in my application, which includes how to modify each table by adding, amending, and deleting. I will also go through how to navigate the program itself.

3.2. Assumptions

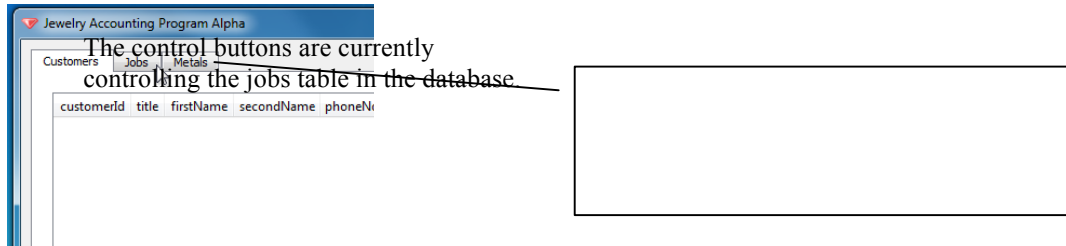
During the tutorial, I go through the basic functionality of the control buttons on the right of the program. I only perform actions on one tab, which is the customers tab but I assume the user can take the use of one tab's control buttons over to the next, because the functionality of the control buttons differs very little on each tab.

I also expect that the user has a very basic understanding of the operating system Windows 7, and seeing as my client has already been confirmed more than capable with Windows 7 applications from my analysis, the tutorial shouldn't be a problem to follow.

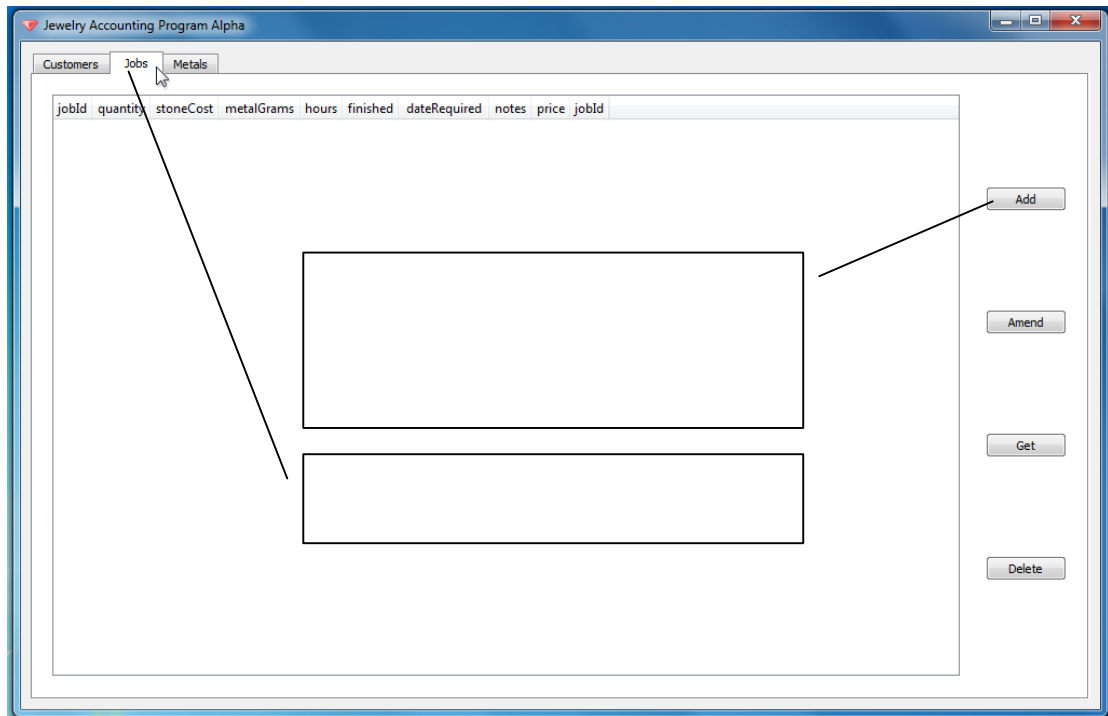
3.3. Tutorial Questions

3.3.1. Navigating the Tabs

Each tab has control buttons for each tab. These buttons allow you to perform changes to the data in the database, and To navigate through the program all you have to do is move the cursor up to the tab bar at the top and click on the tab you wish to navigate to.



Upon clicking, the function buttons on the right will not change, but they will now perform actions on the selected tab's corresponding database table which is indicated by the name of the tab.



3.4. Saving

My application doesn't require any saving, because when an action is performed on the database, it connects with the database, commits the action, saves the database and closes the connection.

This allows the application to close without having save anything.

Enter the data you wish to add to the database. Currently the form contains example data.

You can add to any tab, but in this case

I'm currently adding to the customers tab.

3.3.2. Adding a Record

This button will submit the addition to the database.

This is the add button. Click it to add a

record. To add a record, navigate to the tab you wish to control and click on the add button among the control buttons on the right of the program.

The screenshot shows the 'Jewelry Accounting Program Alpha' window. The 'Customers' tab is selected. The form contains several empty input fields for customer data. On the right side, there are four buttons: 'Add', 'Amend', 'Get', and 'Delete'. An arrow points from the 'Add' button to the form, and another arrow points from the 'Add' button to the text 'This button will submit the addition to the database.'

When clicked, the add form will appear, and you should now enter the data you wish to add to the database, in this case, I have used example data. When you are done, click submit.

The screenshot shows the 'Customer Add Form' dialog box. It contains the following fields with example data entered: Title (Mr), First Name (Example), Second Name (Example), Phone No (1234), Street (Example), Town (Example), County (Example), Postcode (Example), Email (Example), Vat No (123), and Notes (Example). A 'Submit' button is at the bottom. Arrows point from the 'Add' button in the previous screenshot to this dialog box, and from the 'Submit' button to the text 'When you are done, click submit.'

Click "OK" to allow the application to know you acknowledge the addition, confirmation, allowing you to enter more data to add to the database.

Click "OK" when the confirmation comes up.

If you do not wish to add more, click the "X" button to come out of the add form.

The image shows a screenshot of a software application. In the foreground, a 'Confirmation Box' dialog is open, displaying the message 'Customer Added Successfully' and an 'OK' button. Behind it, the 'Customer Add Form' is visible, which contains several input fields: Title (with 'Mr' selected), First Name (with 'Example' entered), Second Name (with 'Example' entered), Phone No (with '1234' entered), Postcode (with 'Example' entered), Email (with 'Example' entered), Vat No (with '123' entered), and Notes (with 'Example' entered). A 'Submit' button is at the bottom of the form. The background shows a table with columns for 'id', 'street', 'town', 'county', 'postcode', 'email', 'vat no', and 'notes'.

You can now either enter in another record to the database, just like before, or you can click the "X" at the top and finish inputting records.

The image shows a screenshot of the 'Customer Add Form' dialog box. The 'X' button in the top right corner of the dialog is highlighted with a red box. The form contains input fields for Title, First Name, Second Name, Phone No, Street, Town, County, Postcode, Email, Vat No, and Notes, along with a 'Submit' button at the bottom. The background shows a table with columns for 'id', 'street', 'town', 'county', 'postcode', 'email', 'vat no', and 'notes'.

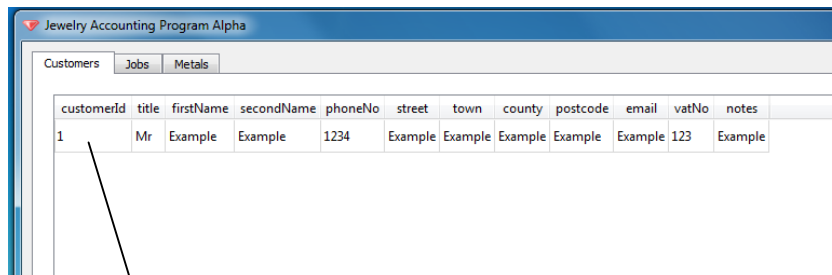
This is the record that was just added.

Elliott Slingsby

Candidate Number: 0836

Centre Number: 225151

Once closed, the application's table will update, showing the record you just added.

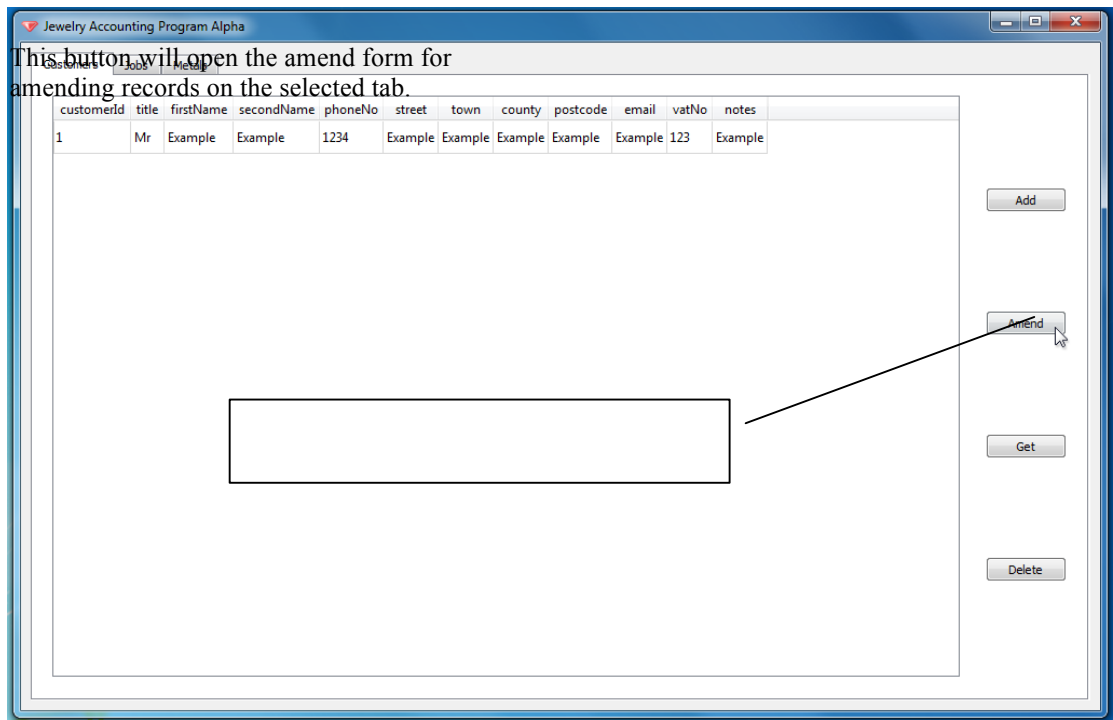


customerId	title	firstName	secondName	phoneNo	street	town	county	postcode	email	vatNo	notes
1	Mr	Example	Example	1234	Example	Example	Example	Example	Example	123	Example

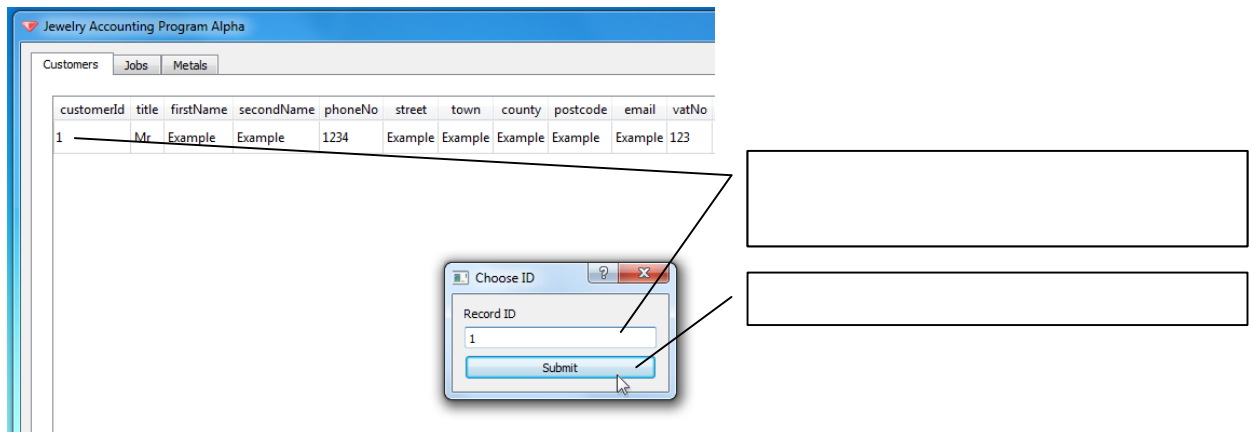
The ID that you enter must correspond to an already existing record, like this one.

3.3.3. Amending a Record

To amend a record, navigate to the tab you wish to control and when you've entered the ID among the control buttons on the right of the program.



Enter the ID of the record you wish to amend. In this case, I have chosen 1. Click submit when you've entered the appropriate ID.



A form similar to the add form should appear. From here you can edit the text boxes to make your amendments. In this case, I have changed the first name. When you are done click submit.

The screenshot shows a 'Customer Amend...' dialog box overlaid on a data table. The dialog box contains the following fields: Title (Mr), First Name (AMENDMENT), Second Name (Example), Phone No (1234), Street (Example), Town (Example), County (Example), Postcode (Example), Email (Example), Vat No (123), and Notes (Example). A 'Submit' button is at the bottom. Annotations include: a line pointing from the 'AMENDMENT' text in the First Name field to a large empty rectangular box; a line pointing from the 'Submit' button to another large empty rectangular box; and a line pointing from the 'AMENDMENT' text to a third large empty rectangular box.

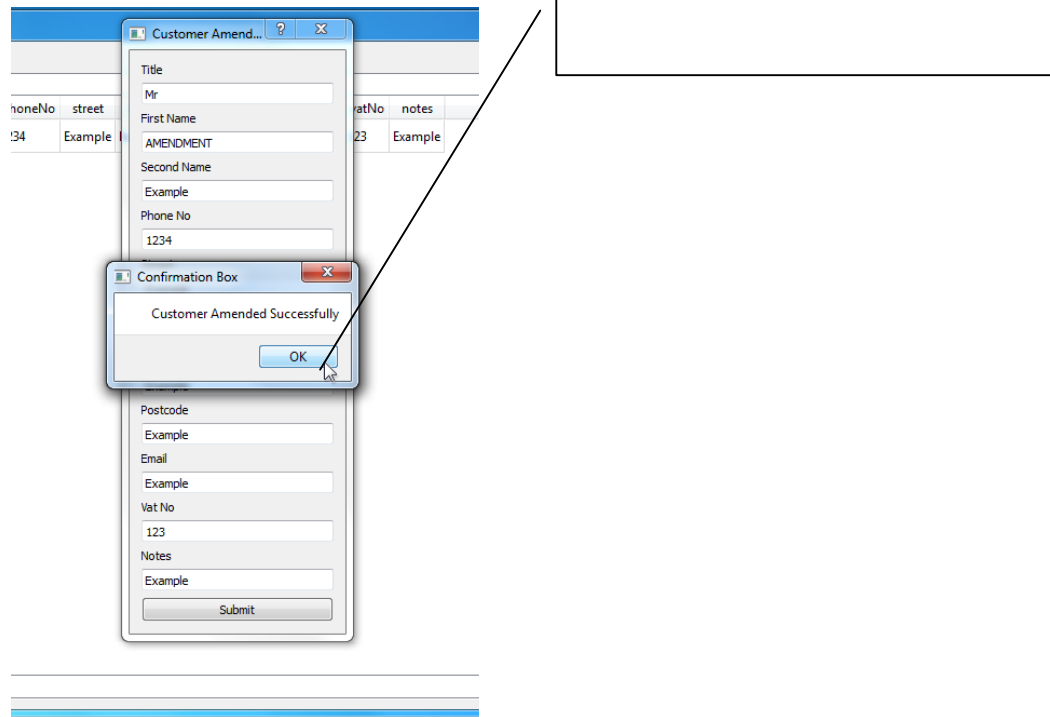
Before continuing, make sure you are amendments are correct and if you see a mistake, click no. If you want to commit your changes, click yes.

This screenshot shows the same 'Customer Amend...' dialog box as before, but with an additional 'jeweler' dialog box in the foreground. The 'jeweler' dialog box asks 'Do you want to commit these amendments?' and has 'Yes' and 'No' buttons. An arrow points from the 'Yes' button to a large empty rectangular box. The 'Customer Amend...' dialog box is partially obscured by the 'jeweler' dialog box.

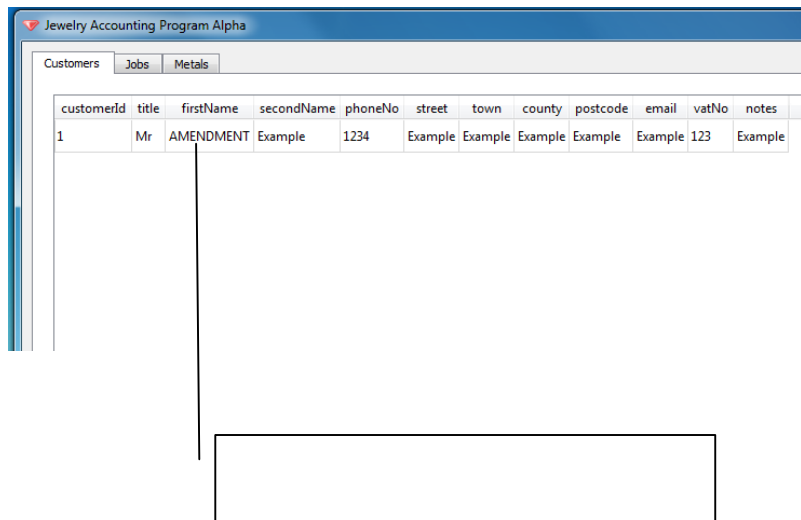
The amendment should now be updated into the database.
 Clicking "OK" let's the application know you acknowledge the changes made.
 Elliott Slingsby Candidate Number: 0836

Centre Number: 225151

Click "OK" when the confirmation comes up.



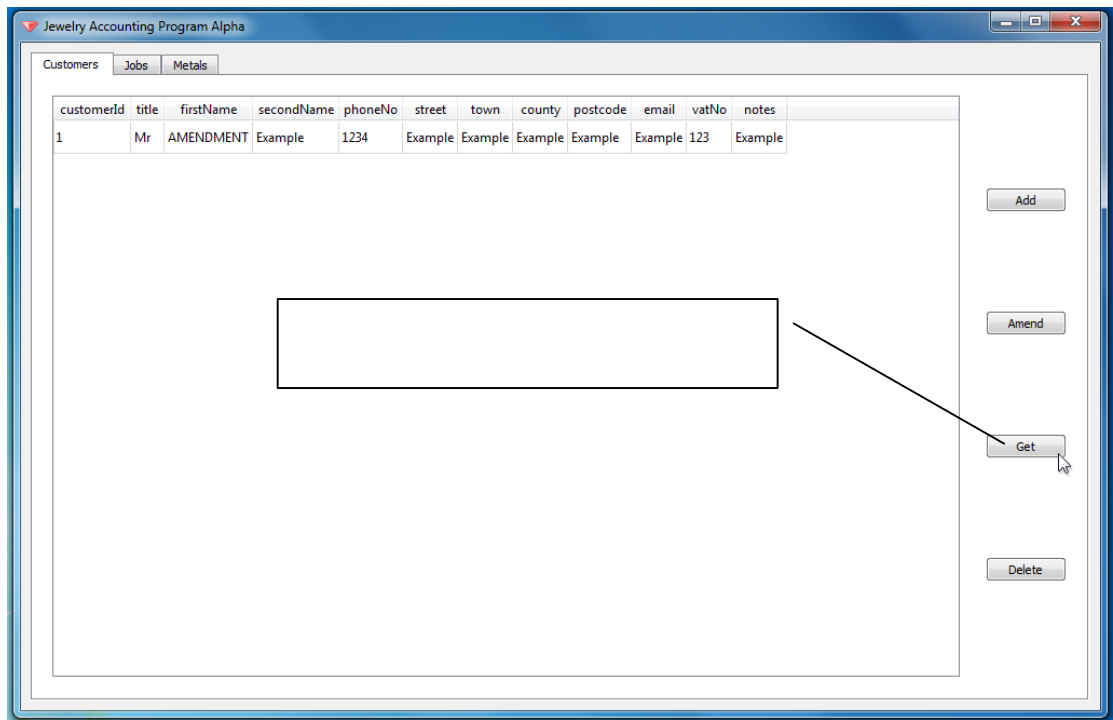
The amend form should then close, and your record should be amended in the database.



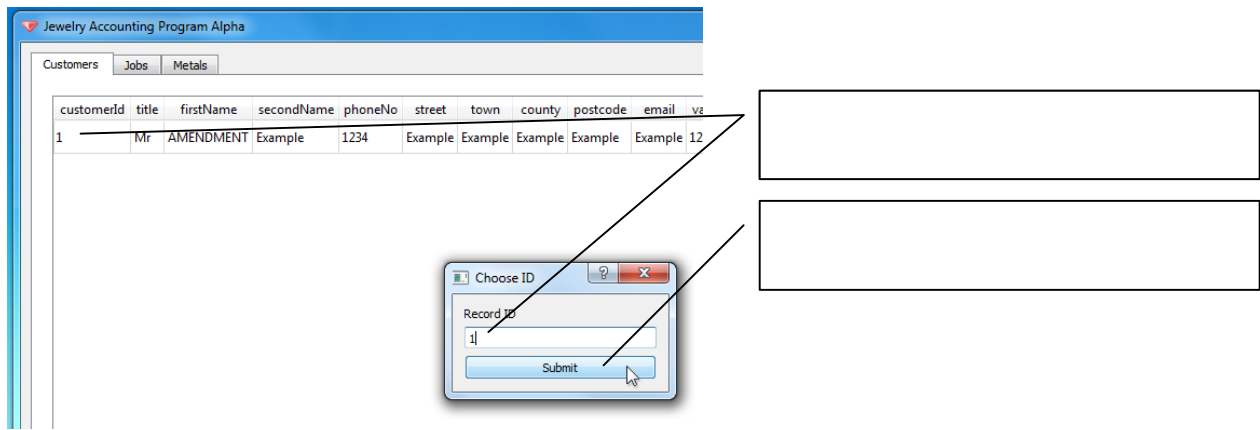
3.3.4. Getting a Record

Click submit when you've entered a valid ID.

To view a specific record in the database, click on the get button among the other control buttons, on the right.



Upon clicking, you should be prompted for the ID of the record you wish to view. In this case, I have used 1.



When you are finished viewing the data, click the "X" button in the top right of the form to close it.
Clicking this button will return you to the main menu.

The screenshot shows a 'Customer Add Form' dialog box. The form has the following fields:

- Customer ID: 1
- Title: Mr
- First Name: AMENDMENT
- Second Name: Example
- Phone No: 1234
- Street: Example
- Town: Example
- County: Example
- Postcode: Example
- Email: Example
- Vat No: 123
- Notes: Example

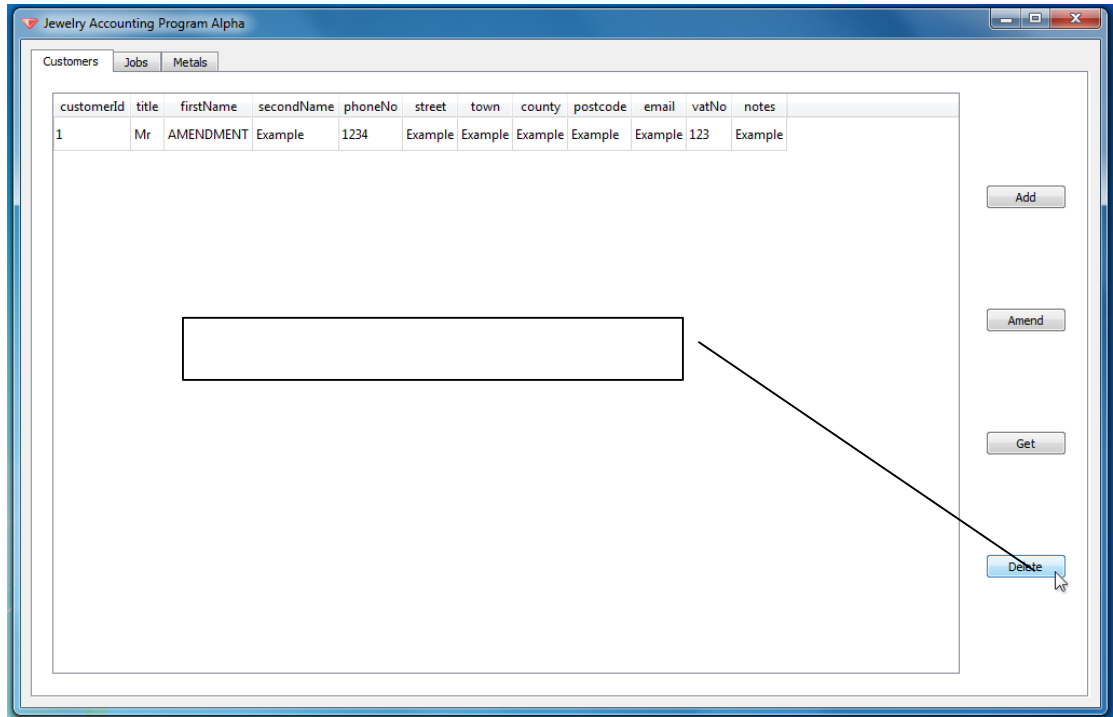
The form is overlaid on a background table. The table has columns: oneNo, street, vatNo, n. The table contains one row with the following data: 14, Example, 123, Example. Two empty text boxes are shown to the right of the form, with arrows pointing to them from the 'X' button in the form's title bar.

The ID you enter should correspond to an already existing record, like this one.

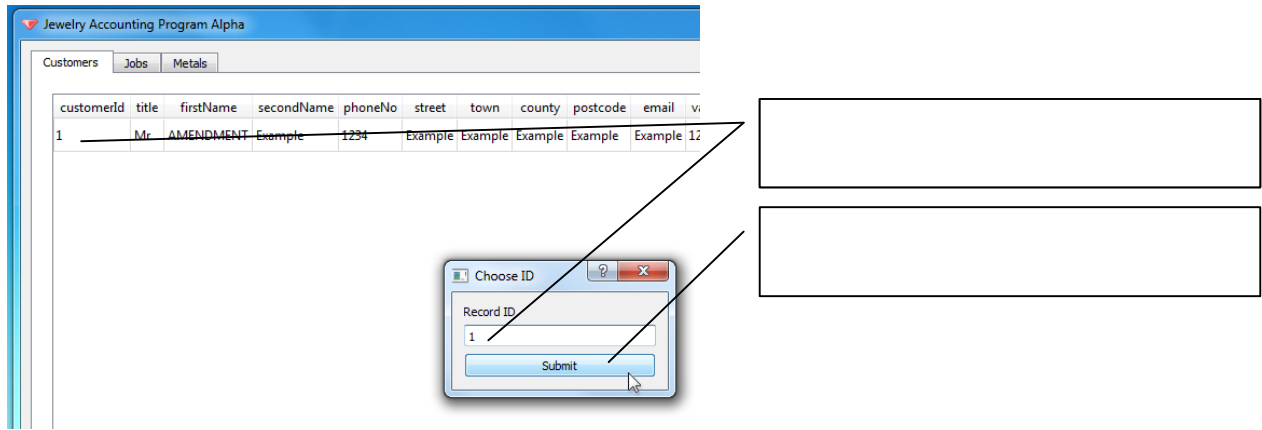
3.3.5. Deleting a Record

Click submit when you've entered a valid ID.

To delete a record, navigate to the tab you wish to control, and click on the delete control button on the right.
To delete a record, click the delete button.

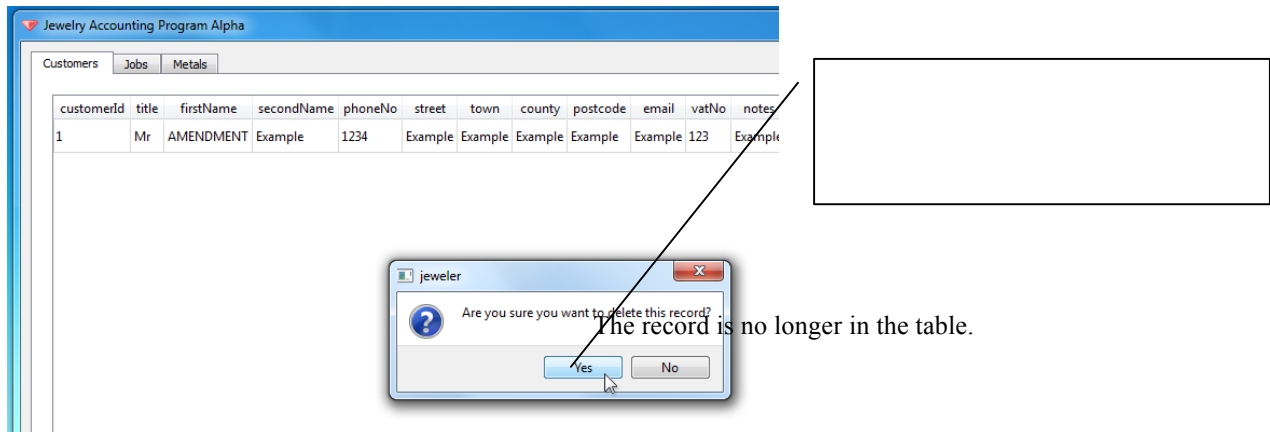


Upon clicking, you should be prompted with the ID of the record you wish to delete. In this case, I have chosen 1. Click submit when you are done.

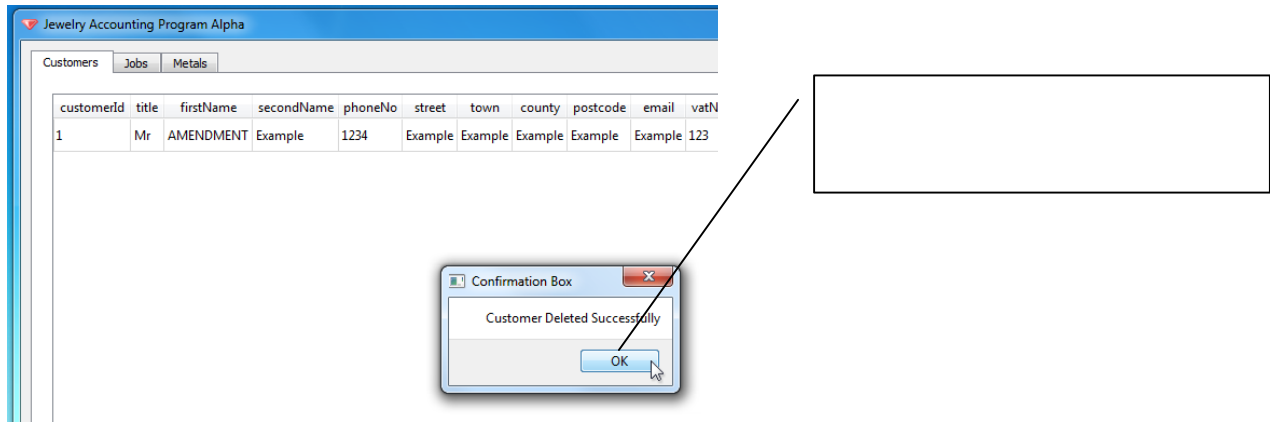


Because deleting a record can be dangerous, you will be asked to confirm the changes. Click yes if you're sure, and no if you're not.

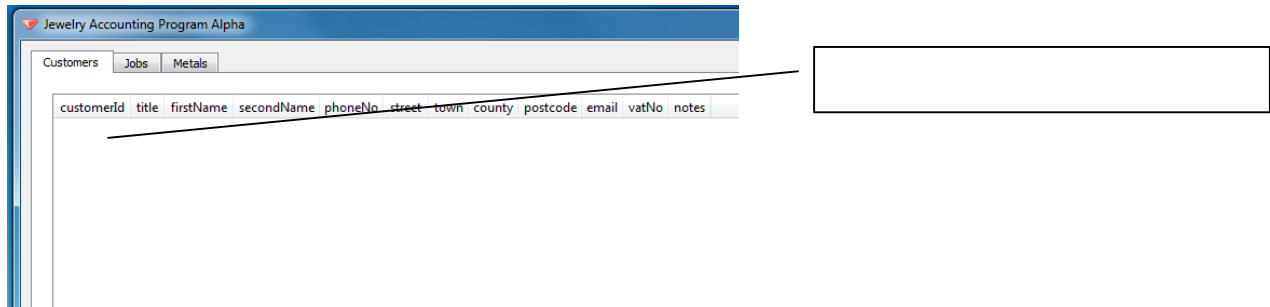
If you are sure you want to delete the chosen record, click yes. If you are unsure, or have changed your mind, click no and the deletion will not be committed.



Click "OK" when the confirmation comes up.



The record you wanted to delete should now be absent from the database.



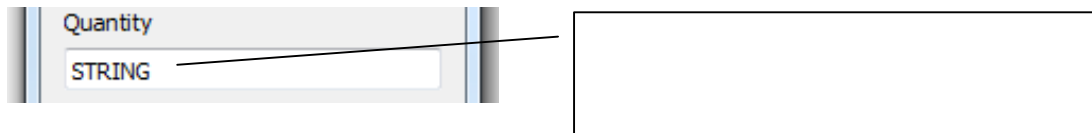
4. Errors

4.1. User error

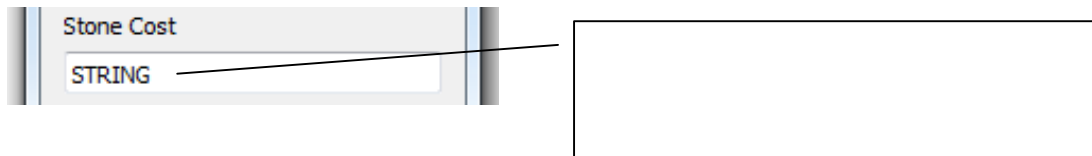
4.1.1. Adding / Amending Jobs

In the adding and amending jobs form, there are multiple text boxes that only allow a certain, unspecified data type to be entered. Otherwise the program will not perform the wanted function. Because all these text boxes suffer from the same error, which is nothing happening when you click submit, examples can't really be shown. Below I will go into each text box that may cause issues when it comes to invalid data entry.

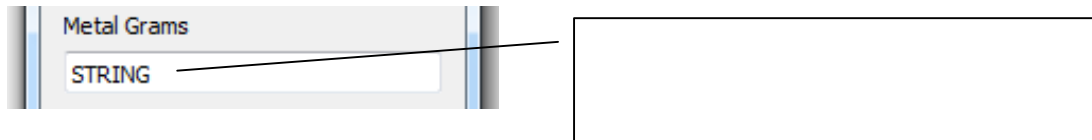
Quantity: This field will only allow integers to be entered. Not doing this will prevent the user from carrying on. An example of how to reproduce this is below.



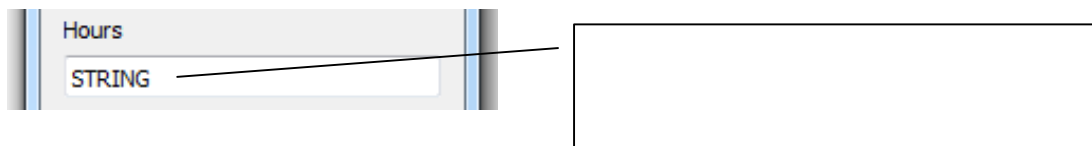
Stone Cost: This field will only allow integers and decimal numbers to be entered. Not doing this will prevent the user from carrying on. An example of how to reproduce this is below.



Metal Grams: This field will only allow integers to be entered. Not doing this will prevent the user from carrying on. An example of how to reproduce this is below.



Hours: This field will only allow integers to be entered. Not doing this will prevent the user from carrying on. An example of how to reproduce this is below.



is entered in place of the
it comes up as True on the

Elliott Slingsby

to cause an error, this would have to be
anything other than a whole number or
decimal number when submitted on any
version of this field

Centre Number: 225151

Finished: This field will only allow integers to be entered, but the database will only allow 0 or 1 to be entered in place of True and False. So if the user enters anything other than a 0 or a 1, it will only come up as True on the get form. An example of how to reproduce this is below.

The screenshot shows a form with a field labeled 'Finished' containing the text 'STRING'. A line points from this field to a large empty rectangular box. Below the form is a table with 10 columns. The first column contains the number '2'. The next four columns contain the value '123'.0'. The fifth column contains the value '123'. The sixth column contains the text 'Broken Finished Field'. The seventh column contains the value '123'. The eighth column contains the value '123'. The ninth column contains the value '123'. The tenth column contains the value '123'. To the right of the table is a small form with a field labeled 'Finished' containing the value 'True'. A line points from the 'Broken Finished Field' column to this small form.

Price: This field will only allow integers and decimal numbers to be entered. Not doing this will prevent the user from carrying on. An example of how to reproduce this is below.

The screenshot shows a form with a field labeled 'Price' containing the text 'STRING'. A line points from this field to a large empty rectangular box.

Customer ID: This field will only allow integers to be entered. Not doing this will prevent the user from carrying on. An example of how to reproduce this is below.

The screenshot shows a form with a field labeled 'Customer Id' containing the text 'STRING'. A line points from this field to a large empty rectangular box.

None of these errors will cause any fatal damage to the database, nor will it make the program freeze or terminate, they simply just prevent the user from carrying on and committing any changes.

4.1.2. Adding / Amending Metals

Like the job's adding and amending forms, the metals adding and amending forms also contain a text box that may cause issues if the user enters invalid data. There is only one text box that has a specific data type.

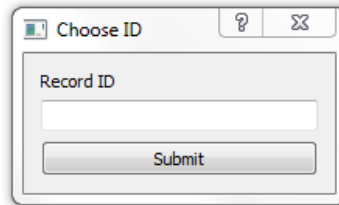
Cost Per Gram: This field will only allow integers and decimal numbers to be entered. Not doing this will prevent the user from carrying on. An example of how to reproduce this is below.

The screenshot shows a form with a field labeled 'Cost Per Gram' containing the text 'STRING'. A line points from this field to a large empty rectangular box.

Because these cause the same error as the fields in the job's adding and amending forms, these also will not cause the program to freeze or terminate.

4.1.3. Choosing IDs

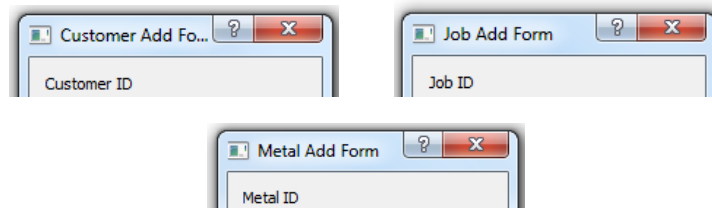
Whenever the user wishes to perform a control function on the database that requires a target record's ID to entered, the choose ID box will appear. Because this is the same instance whenever it appears, it suffers from the same error on every tab. Below you can see the box.



Whenever the user enters an ID that points to a non-existing record, the box will suddenly close, giving no indication of an error. This doesn't cause any long-term damage, and it will not cause the program to crash or freeze.

4.2. Program bugs

The only program bugs that managed to make it into the final application were the miss labelling of the get forms on each tab. Instead of title "Customer Get Form", the actual form window title reads "Customer Add Form". Examples of which are below.



Fortunately this is a minor program bug which could easily be amended in the code, and shouldn't cause any harm to the usability of the application.

4.3. Incomplete sections of code

My application doesn't contain any obvious incomplete sections of code, because none of the incomplete sections ever got implemented into the GUI.

For example, there is an entire functioning controller for a non-existent invoice table in the database, but it's functionality never got implemented into the GUI, so it should cause no problem for the average user.

```
set /p Drive="Drive Letter >>> "  
XCOPY "application.db" %Drive%\data  
PAUSE
```

5 System Recovery

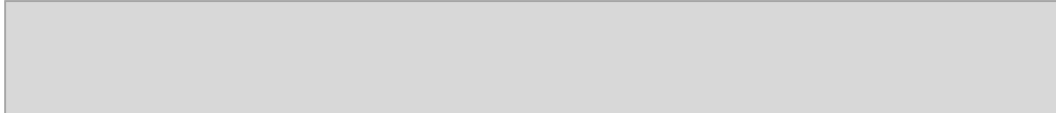
Clicking on this, in the right click menu, will open an explorer window of the directory the shortcut points to.

5.1. Backing up data

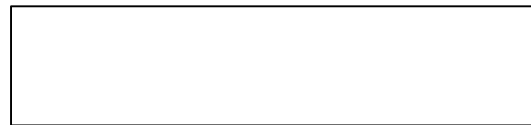
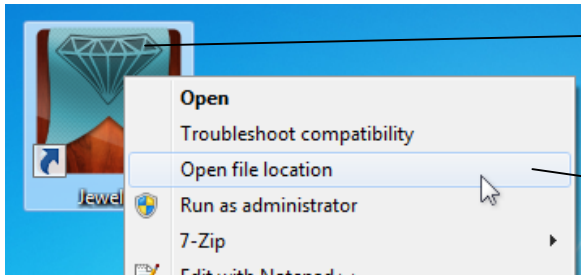
To backup my program, the user will need a few spare USB pen drives. It is suggested that every month the database is backed up onto one of the pen drives, and stored in a safe location.

The only data that requires backing up in the application is the database file itself. Because this is quite a simple problem to solve, I thought that creating a Batch file that performs a simple XCOPY to a chosen drive letter would suffice. Below you can see the batch code.

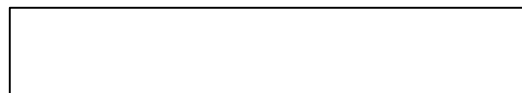
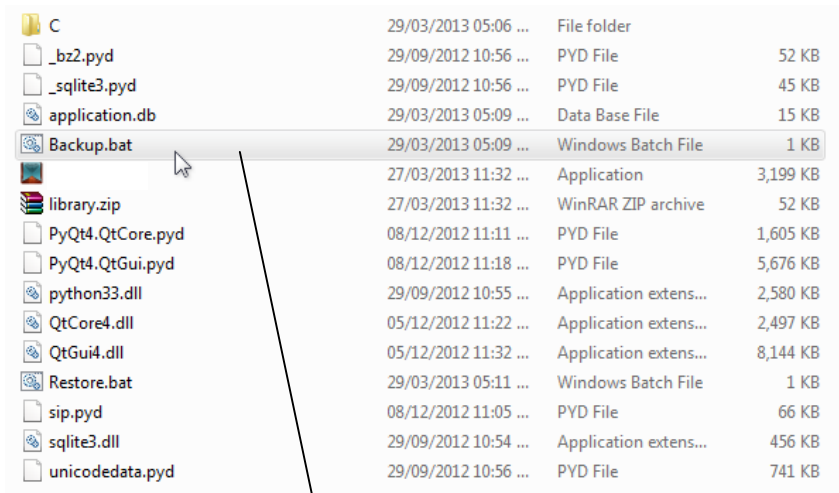
Inside the directory is the Backup.bat file, which allows for easy backing up.



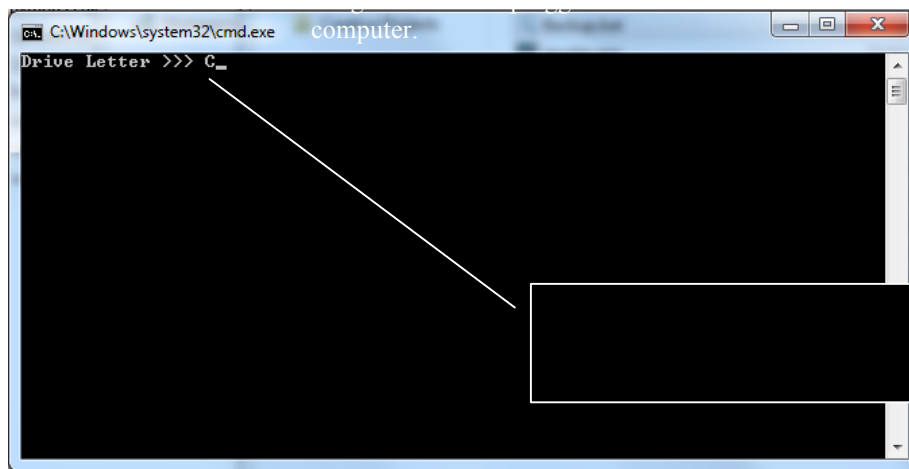
To easily locate the backing up batch file, you can simply right click on the shortcut made during installing and click "open file location".



Once in the directory, locate "Backup.bat", and double click on it to run it.



Upon running the batch script, you will be prompted for the drive letter of the USB you wish to back up to. To locate this, you can go to my computer and see the USB device under "Devices with Removable Storages". I currently do not own a USB stick, so for this example I have used my main hard disk drive, which has a drive letter of C.



When you press enter after entering the drive letter, a confirmation message will appear. The user then needs to press any key to close the script.



```
@ECHO OFF
set /p Drive="Drive Letter >>> "
jwellerPY %Drive:~0,1%":\application.db" "application.db"
PAUSE
```

Elliott Slingsby

Candidate Number: 0836

Centre Number: 225151

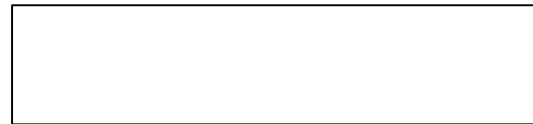
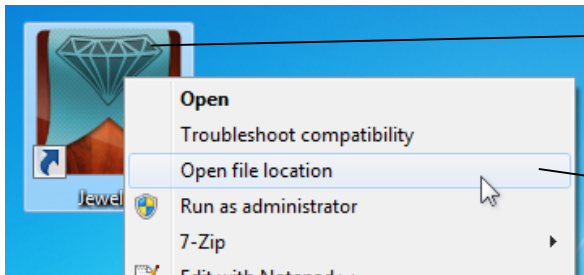
5.2. Restoring data

To restore my program, the user will need the pen drives they used to previously back-up on.

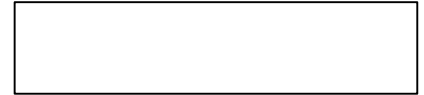
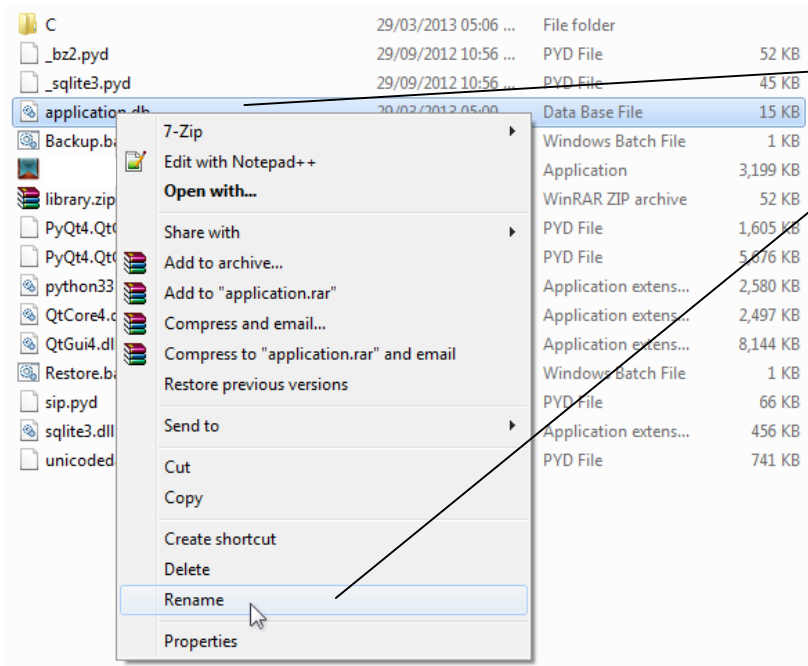
Like backing up, I also created a batch script that performs a simple XCOPY, except this one works in reverse. Meaning it copies the database from the USB drive, to the program. Below is the batch script.



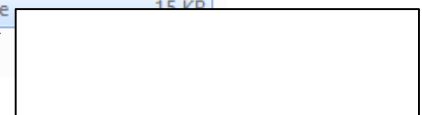
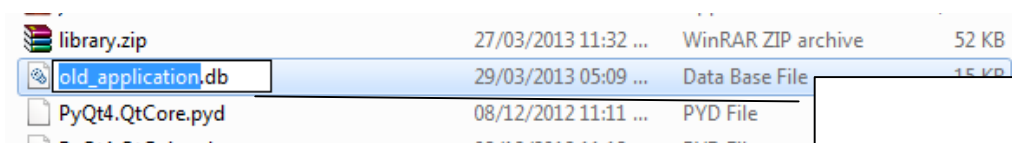
To easily locate the backing up batch file, you can simply right click on the shortcut made during installing, and click "open file location".



When the directory opens, locate the database file "application.db", right click on it, and select rename.

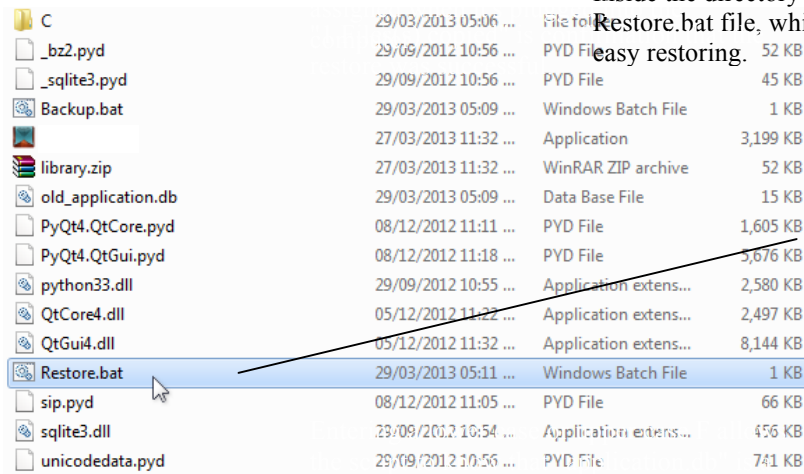


Rename the file to something different, I suggest adding the prefix "old_". This is in case something is wrong with the restored database. If there is, delete it, and rename the old one back to "application.db".

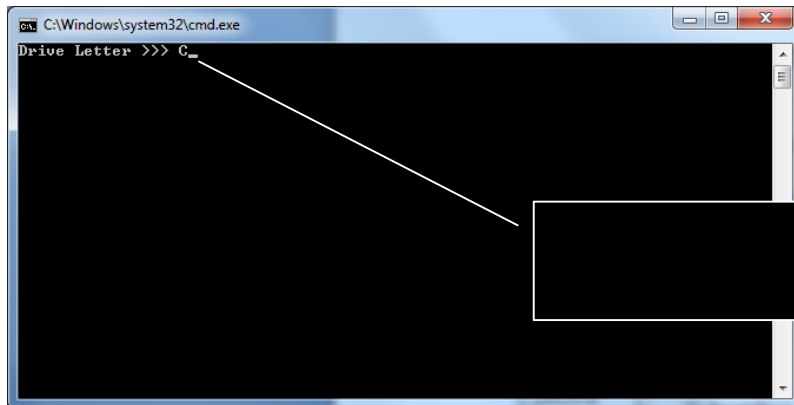


Once in the directory, locate "Restore.bat", and double click on it to run it.

Inside the directory is the
Restore.bat file, which allows for
easy restoring.



Upon running the batch script, you will be prompted for the drive letter of the USB you wish to restore from. To locate this, you can go to my computer and see it under "Devices with Removable Storages". I currently do not own a USB stick, so for this example I have used my main hard disk drive.



You will be prompted to specify if "application.db" is a file or folder, you should select "F" for file. When you press F, the script should restore the database, and you can press any key to close the window.

