

PROJECT DOCUMENTATION

PROJECT AIM

To create a software information system that stores the customers, orders and the menu of a pizza restaurant and produces statistics from this. It allows a customer to input their personal details and make an order, and also allows restaurant employees to edit, add or delete items in the database. **Qt Designer** was used to create a simple, clear user interface, so that the user could easily follow instructions, minimising user error.

This project focuses mainly on having a fully functioning program with minimal bugs and errors. If more time was available more features would be added to continue improving the program.

Project Structure

The project structure is simple and intuitive so should the program need to be modified, anyone can open the project and know how to find the file that needs changing. The qt designer files are in one file and the converted qt files are in **src/gui/qt_py** folder. Anything to do with the program interface is in the **src/gui** folder so that similar python files are grouped together. The binary search tree classes are in the **src/util**.

The functionality of the project is split into two separate parts. The customer interface and the employee interface. Each has its own source code file in **src/gui**.

Storage Backend

The principle way of storing data is using an excel sheet (**data.xls** contained within the **data_storage** folder). Customers and employees don't directly access the excel file while using the main program, this will reduce any bugs in the program if the excel file is not filled in correctly, tampered with, etc.

The only time the file will have to be directly accessed is when adding/editing/deleting employee logs, but this can be done by the restaurant manager that has been trained to do so.

The excel file has a menu sheet. This contains the menu items and their corresponding prices. This information is then stored as a binary search tree in the menu class (**menu.py**). This is so prices can be very quickly obtained during the program, making the program more efficient.

Non-Standard Packages used

```
• from xlutils.copy import copy
• import xlrd
• import datetime
• import unicodedata
• from PySide.QtCore import *
• from PySide.QtGui import *
• import sys
```

xlutils.copy: allows you to copy a spreadsheet

xlrd: reads and writes to an excel file
datetime: allows you to find date and times
unicodedata: is used to convert Unicode to ascii
Remaining packages are used for any qt window classes

User Manual

How to Run the Program

The user must go into the **run_program.py** file located in:

Lidia Dynes Martinez\src\gui\run_program.py

Then simply run this file and the program interface will show. From this point simply follow the instructions written in the interface. To get into the employee interface you need to login (**the login is case sensitive**).

YOUR EMPLOYEE LOGIN IS: pizzapalacePETAR

Overview of program interface

Qt Designer was used to make the program interface. Each window has its own class that class different methods depending on which buttons are pressed by the user. These classes are all linked together by using various methods. In the qt classes, other classes are also instantiated depending on what information is input into the interface. For example, in the new customer window a customer is made using the imported customer class. This is then passed through the various window classes so that it can be used elsewhere and the order can be linked to the customer later on.

The flowchart below shows how the overall structure of the program and how the user can access different parts of the interface. The Customer goes down one path and cannot access the other and the employee can access the important employee interface using a login.

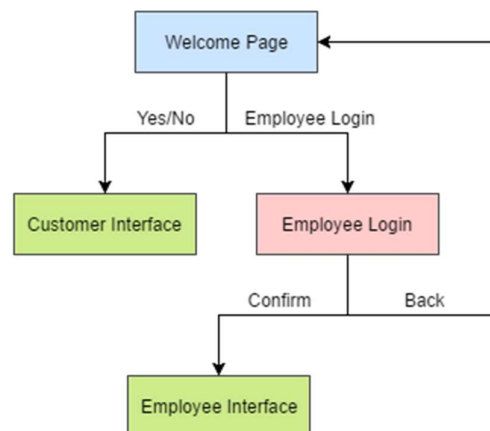
Flow Charts Key

Blue = window where user only interacts with push buttons

Red = window where user inputs information

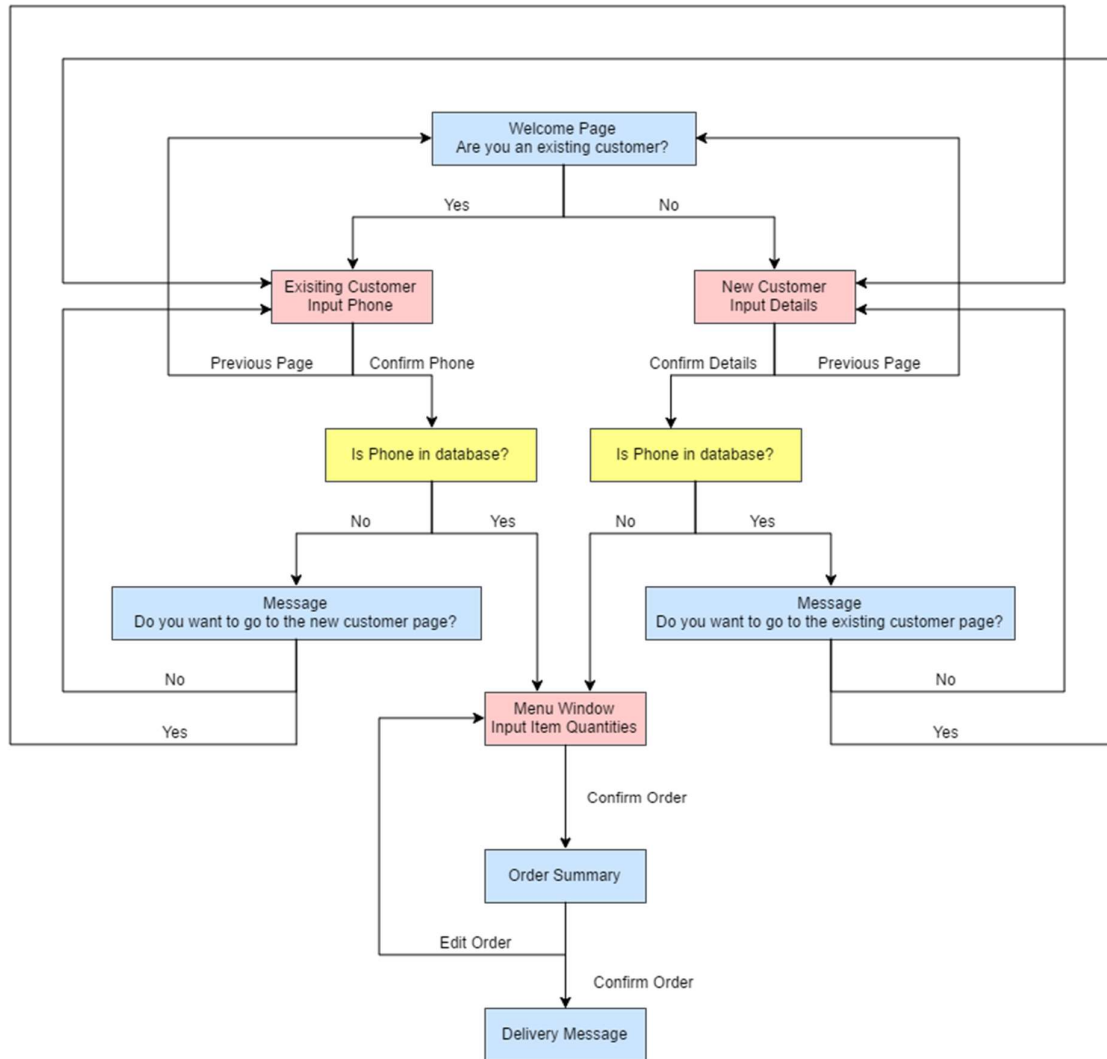
Yellow = behind the scene program logic

Green = Interface



Customer Interface Flowchart

This flow chart describes the process the customer goes through in the interface in order to complete an order. It also shows how the customer can go back and edit information if they realise it's wrong by using the push buttons at the bottom of each screen



How to edit information using the employee interface

After running the program, the welcome page pops up. In the top right hand side of the window there is an employee login push button which leads to a login page. The employee can then login and it will take them to the employee interface.

There are four tabs, the first three allow you to edit, delete or insert items in the respective fields. The last tab just shows a table with the restaurant statistics and is not editable. To make changes on the each of the first three tabs the employee must do the follow the steps below correctly and therefore the employee requires some training.

TabWidget

Customers Orders Menu Statistics

	First Name	Last Name	Postcode	
1	lidia	dynes	gl516dz	078250
2	fraser	price	sw66tj	078524
3	tessa	smulders	w87dj	079420
4	anna	bernbaum	sw69fk	074168
5	shiv	bhatnagar	sw78qw	074163
6	Tom	Smith	W8 7BD	077852
7	Barry	Thompson	W6 82F	079204

Add Customer Delete Customer

Enable Editing

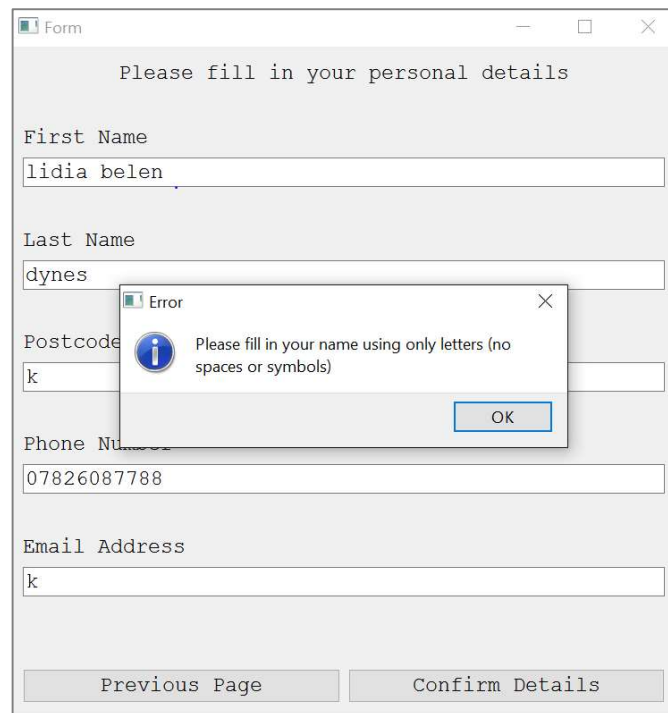
- To be able to edit the information in the table or use the delete/add button the **'Enable Editing'** must be pressed as the delete/add button are initially greyed out
- To delete a row, click on the desired row and press the delete button
- To add a row, click on the row above where you want to insert the row and press the add button
- Once a row is added each cell in the row must be completed with the correct information
 1. names are all strings with no spaces or symbols
 2. phone numbers are a single, 11-digit integer
 3. the list of orders, total orders and money spent are filled correctly with only numbers
 4. dates and times should be filled in before saving
- To **save** the changes, press the **'Save Information'** button (This will also update the statistics tab)

Things to bear in mind

Customer Interface

Once the interface pops up, the instructions are fairly clear in terms of what the customer has to do in order to complete the order. At each stage the information that is written in the

interface by the customer is checked to prevent introducing any bugs to the program. For example if the user has two first names an error message will pop up and notify the user that it is not a valid input.



The image shows a software window titled 'Form' with a close button. Inside, the text 'Please fill in your personal details' is centered. Below this are five input fields: 'First Name' (containing 'lidia belen'), 'Last Name' (containing 'dynes'), 'Postcode' (containing 'k'), 'Phone Number' (containing '07826087788'), and 'Email Address' (containing 'k'). At the bottom are two buttons: 'Previous Page' and 'Confirm Details'. An 'Error' dialog box is overlaid on the form, displaying an information icon and the message: 'Please fill in your name using only letters (no spaces or symbols)'. The dialog has an 'OK' button.

If a customer chooses the button 'No' on the welcome page, it means they think they are an existing customer. When they input the phone number in the next window the program checks that that number is in fact already in the database to avoid any errors. If the number is not found in the database the message below pops up and gives the customer the option to retype the number or go to the new customer page instead.

The image shows a screenshot of a software application window titled "Form". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area of the window displays a message: "01212526578 is not in our database. Would you like to go to the existing customer page?". Below the message are two buttons: "Yes" and "No". At the bottom of the window, there are two more buttons: "Previous Page" and "Confirm Number".

Employee Interface

In a similar way to the customer interface, the login the employee uses is checked against possible logins in the excel sheet and if the input login is wrong an error message will pop up notifying them of the error.

There was not enough time to add this safety feature to the employee interface; in order to use the program the employee should therefore be trained so they don't make errors when editing information and saving it to the excel database. To improve the interface, message boxes would be implemented so that if the employee filled out any information wrong they would pop up and notify the user. In this way you would avoid entering any erroneous data into the database and prevent bugs forming.