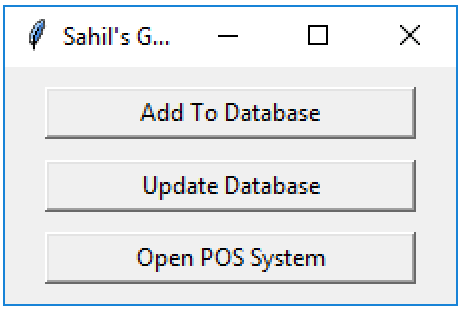
Output

Startup Window



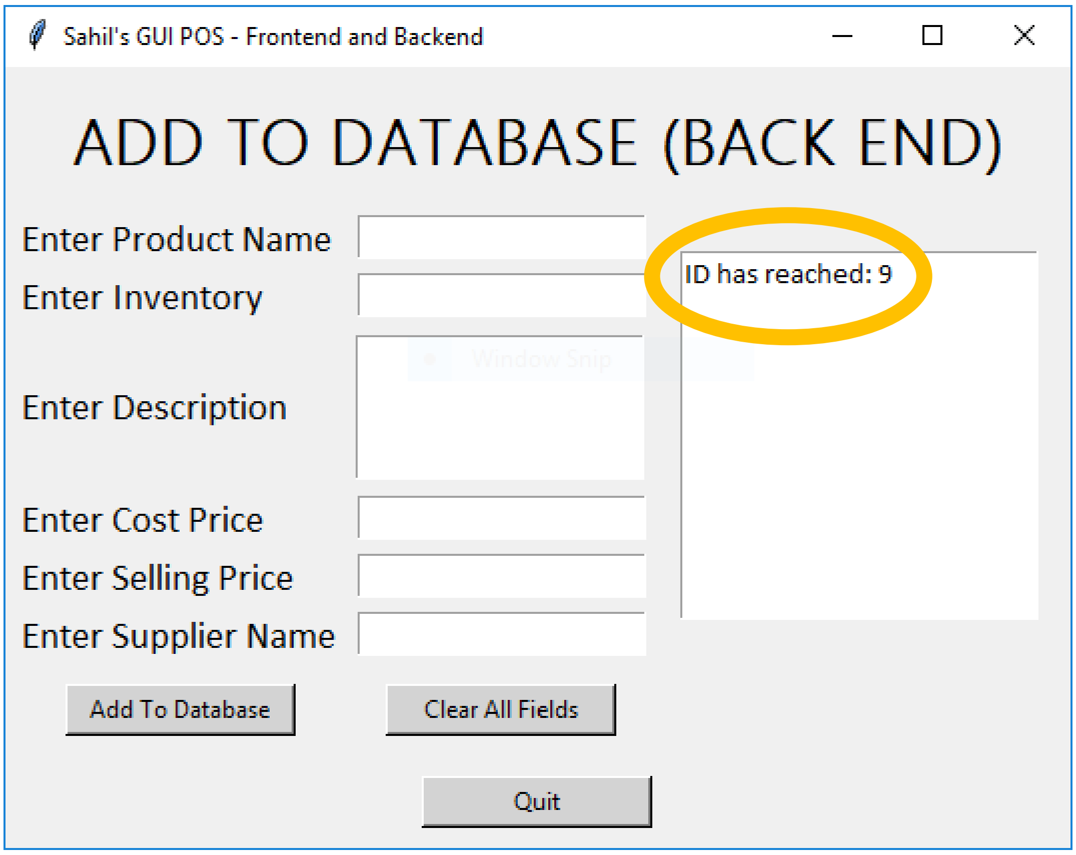
Startup Window

Startup window displays three buttons to navigate to specific sub-programs, namely:

1. Program 1 - Add to Database
2. Program 2 - Update Database
3. Program 3 - Open POS System

Output

Program 1- Add to Database


Sahil's GUI POS - Frontend and Backend

ADD TO DATABASE (BACK END)

Enter Product Name

Enter Inventory

Enter Description

Enter Cost Price

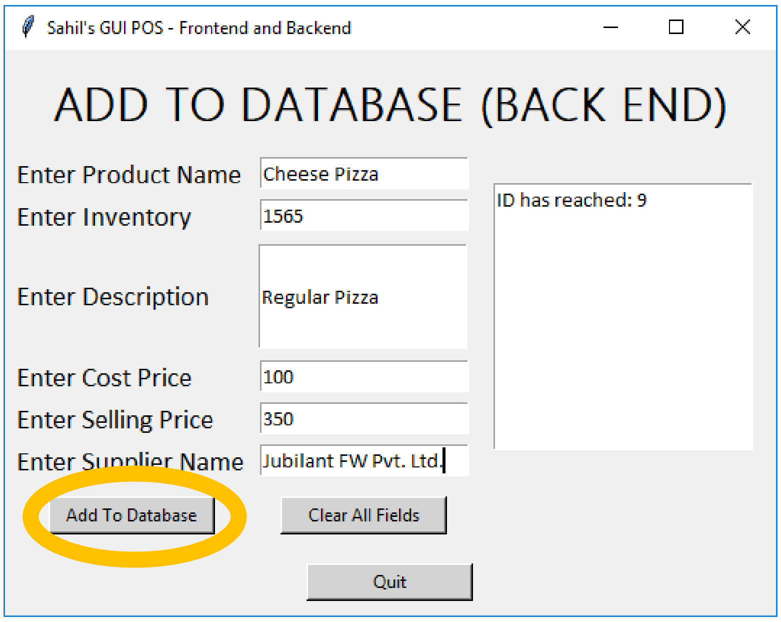
Enter Selling Price

Enter Supplier Name

ID has reached: 9

This sub-program adds inserted values into MySQL Database ‘store’ table ‘inventory’ by fetching (get()) required values from entries.

*Highlighted ID value reflects the last Primary Key (ID) from the ‘inventory’ table.


Sahil's GUI POS - Frontend and Backend

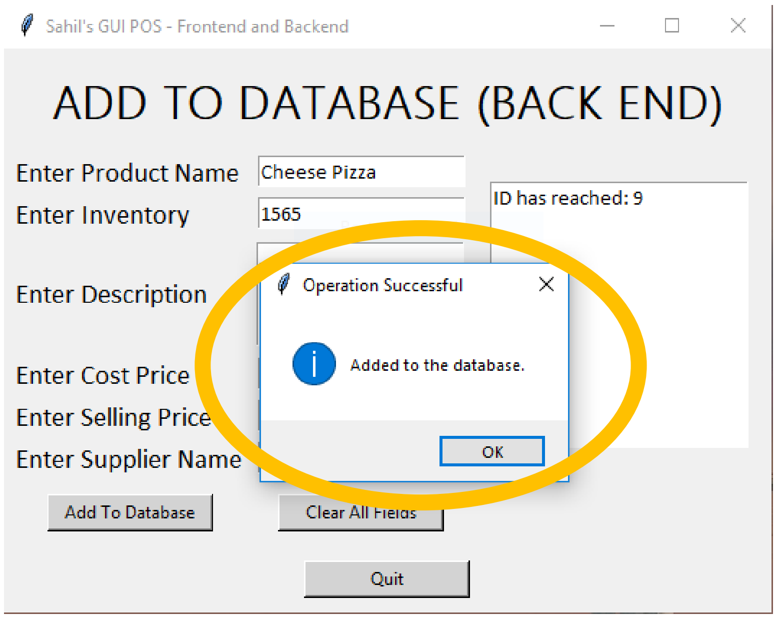
ADD TO DATABASE (BACK END)

Enter Product Name	Cheese Pizza
Enter Inventory	1565
Enter Description	Regular Pizza
Enter Cost Price	100
Enter Selling Price	350
Enter Supplier Name	Jubilant FW Pvt. Ltd.

ID has reached: 9

Output

Values can be inserted in the following way.


Sahil's GUI POS - Frontend and Backend

ADD TO DATABASE (BACK END)

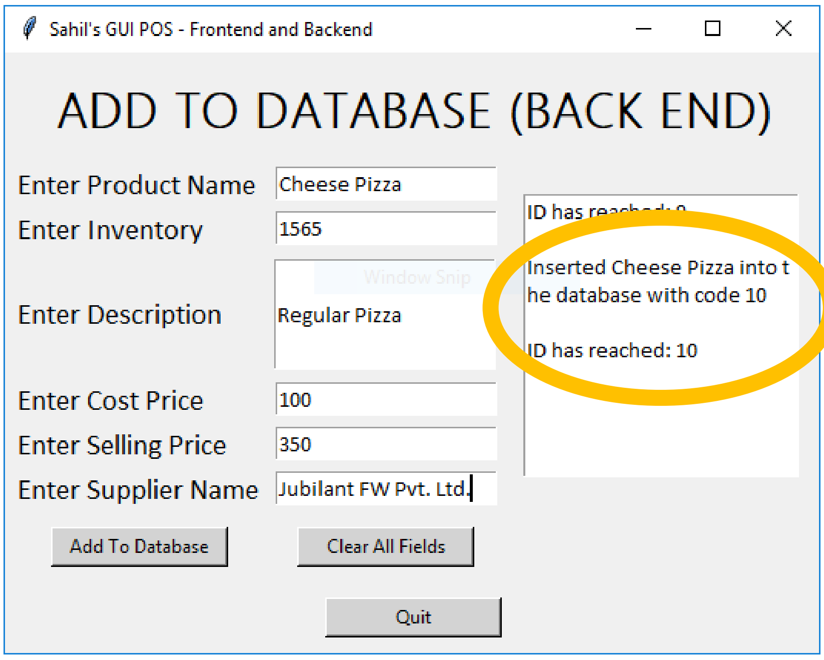
Enter Product Name: Cheese Pizza
Enter Inventory: 1565
Enter Description
Enter Cost Price
Enter Selling Price
Enter Supplier Name

ID has reached: 9

Operation Successful

Added to the database.

**Highlighted: tkinter.messagebox* - UI Element


Sahil's GUI POS - Frontend and Backend

ADD TO DATABASE (BACK END)

Enter Product Name: Cheese Pizza
Enter Inventory: 1565
Enter Description: Regular Pizza
Enter Cost Price: 100
Enter Selling Price: 350
Enter Supplier Name: Jubilant FW Pvt. Ltd.

ID has reached: 9

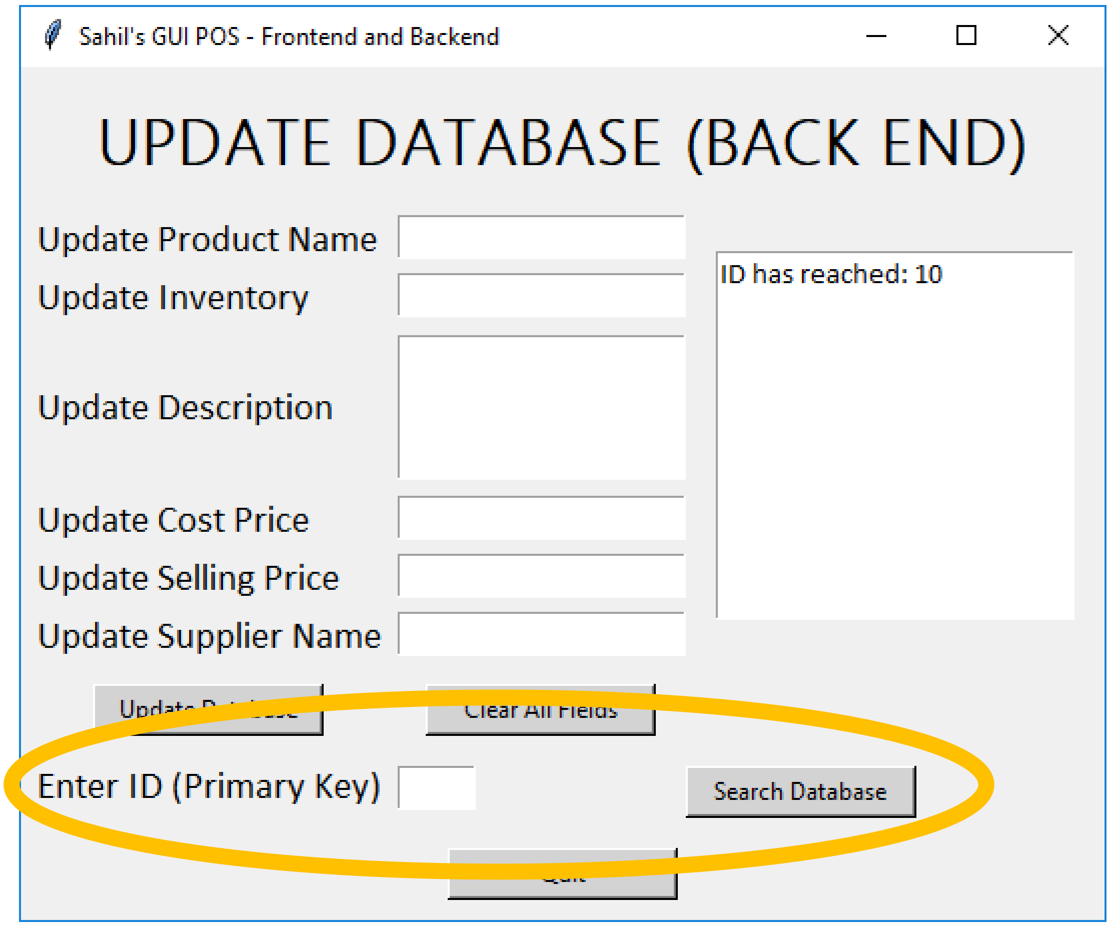
Inserted Cheese Pizza into the database with code 10

ID has reached: 10

*Highlighted: Informing user that ID in ‘inventory’ table has reached value 10 and that product has been successfully inserted into the database.

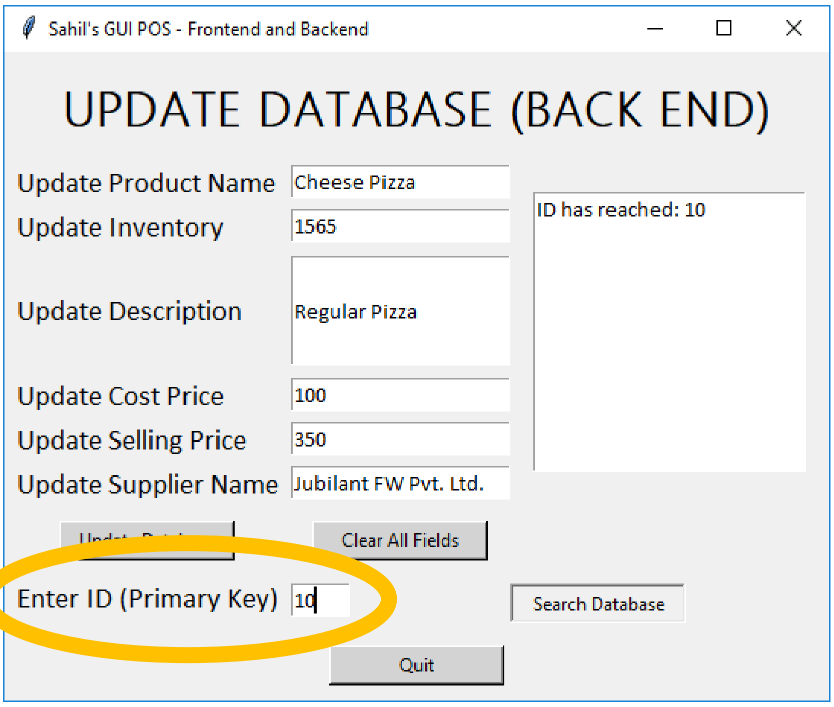
Output

Program 2- Update Database



This sub-program fetches values from MySQL database *store.inventory* and displays them in the given entries to be updated.

Output


Sahil's GUI POS - Frontend and Backend

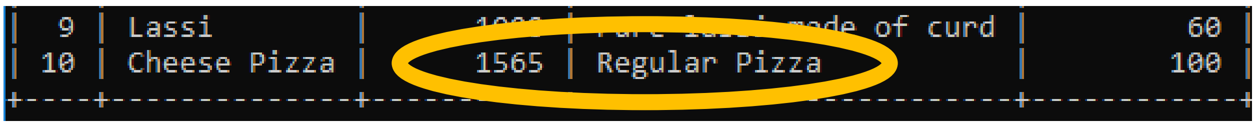
UPDATE DATABASE (BACK END)

Update Product Name: Cheese Pizza
Update Inventory: 1565
Update Description: Regular Pizza
Update Cost Price: 100
Update Selling Price: 350
Update Supplier Name: Jubilant FW Pvt. Ltd.

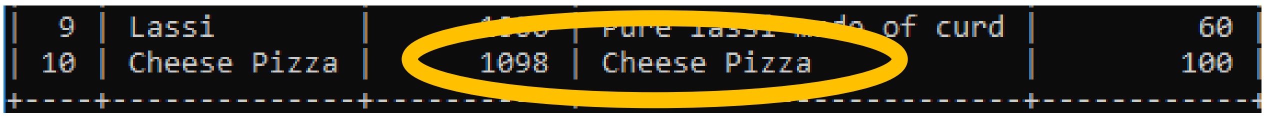
ID has reached: 10

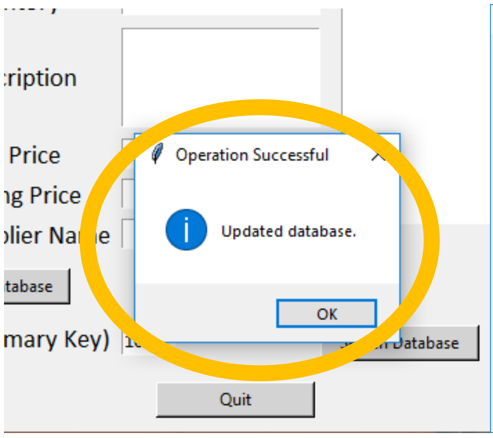
Enter ID (Primary Key): 10

Database searched for Primary Key ‘10’ **(as was added using Program 1)** and values were reflected correctly as shown.

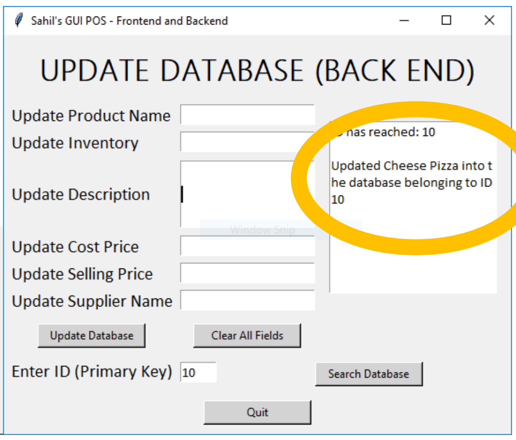

```
|  9 | Lassi        |   [illegible] | Pure lassi made of curd |           60 |
| 10 | Cheese Pizza |   1565 | Regular Pizza           |          100 |
+----+--------------+--------+-------------------------+--------------+
```

Here description for ‘Cheese Pizza’ was changed from ‘Regular Pizza’ to ‘Cheese Pizza’, and stock quantities were updated, as is reflected in the MySQL Query.


```
|  9 | Lassi        |   [illegible] | Pure lassi made of curd |           60 |
| 10 | Cheese Pizza |   1098 | Cheese Pizza            |          100 |
+----+--------------+--------+-------------------------+--------------+
```


Operation Successful

Updated database.


Sahil's GUI POS - Frontend and Backend

UPDATE DATABASE (BACK END)

Update Product Name
Update Inventory
Update Description
Update Cost Price
Update Selling Price
Update Supplier Name

ID has reached: 10

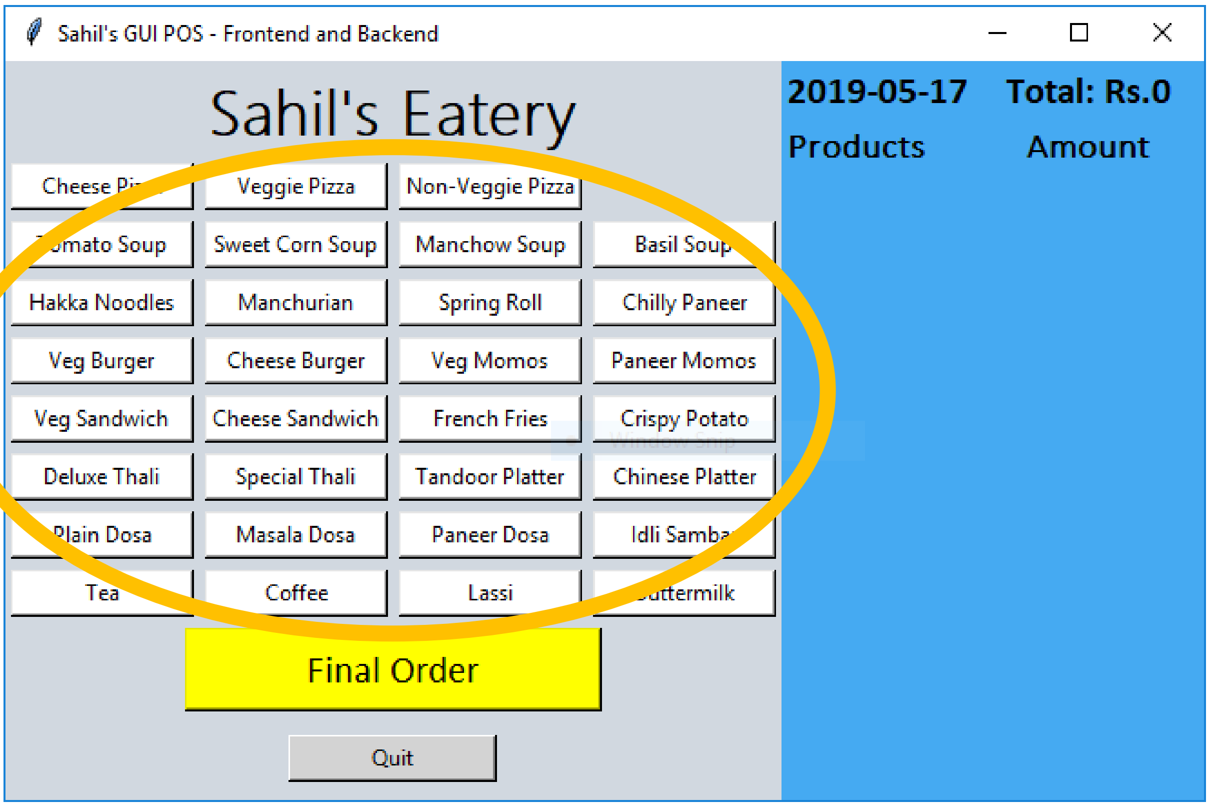
Updated Cheese Pizza into the database belonging to ID 10

Enter ID (Primary Key): 10

**Highlighted: tkinter.messagebox* - UI Element & Textbox updated for UI

Output

Program 3- POS System


Sahil's GUI POS - Frontend and Backend

Sahil's Eatery

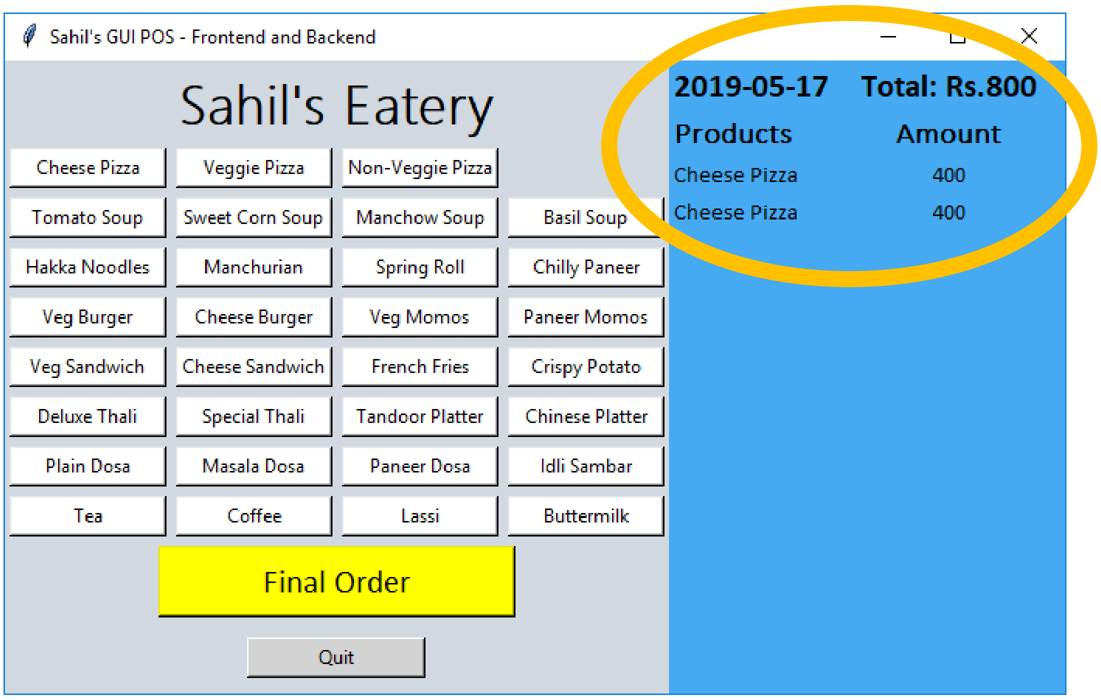
2019-05-17 Total: Rs.0

Products Amount

*Highlighted: Order buttons for ease of use and speed

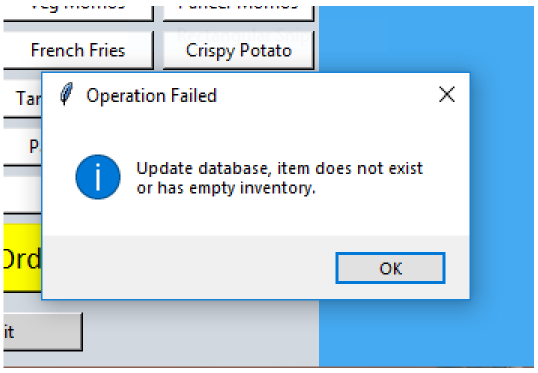
This sub-program is the main feature, it creates invoices, calculates change, updates the 'transactions' database, updates inventory from 'inventory' table and creates a text invoice.

Output



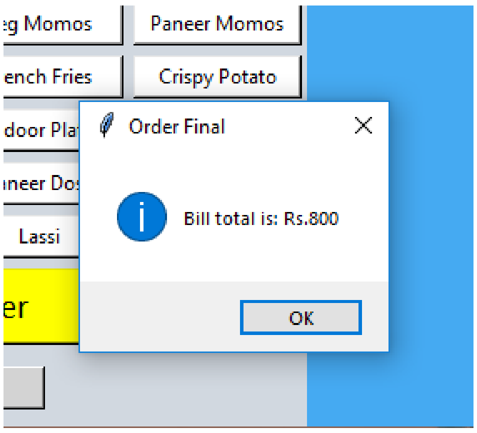
```
2019-05-17    Total: Rs.800
Products          Amount
Cheese Pizza       400
Cheese Pizza       400
```

*Highlighted: Added items shown in right frame along with total order amount



Update database, item does not exist or has empty inventory.

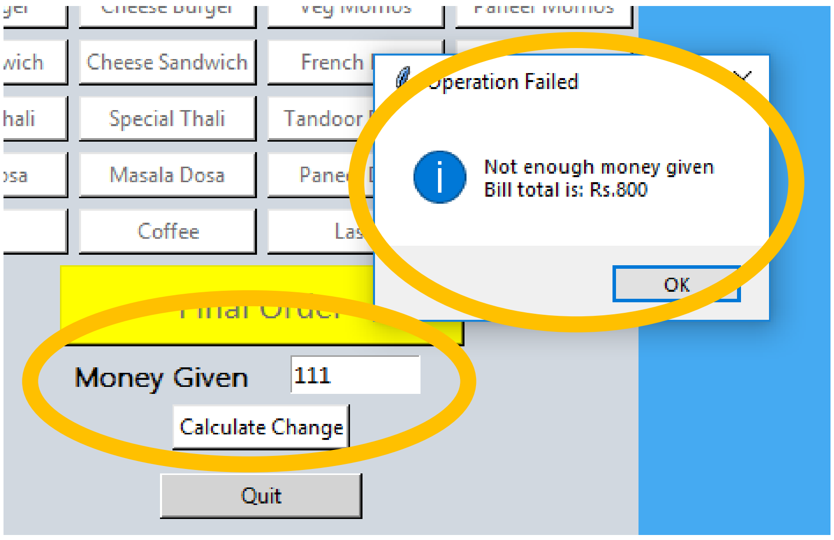
If order button pressed and item does not exist, or has zero stocks, message is displayed.



Bill total is: Rs.800

Final Order button pressed, *tkinter.messagebox* displayed and order buttons are **DISABLED**

Output

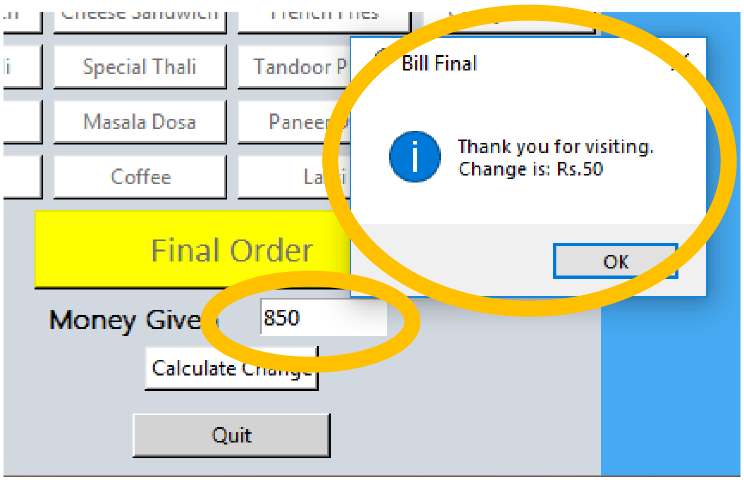


Operation Failed

Not enough money given
Bill total is: Rs.800

111

*Highlighted: As *Final Order* pressed, all order buttons disable and fade, dynamically showing change calculator, if money given lower than order, *tkinter.messagebox* displayed saying money not enough.

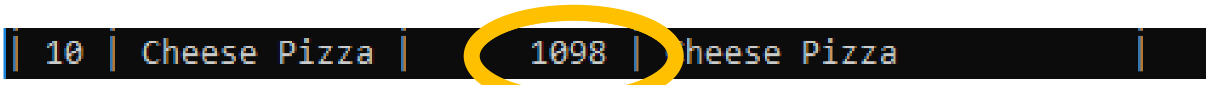


Bill Final

Thank you for visiting.
Change is: Rs.50

850

*Highlighted: Given amount more than or equal to Total Bill, *tkinter.messagebox* displays change requires, order is now final.



```
| 10 | Cheese Pizza |      1098 | Cheese Pizza          |
```

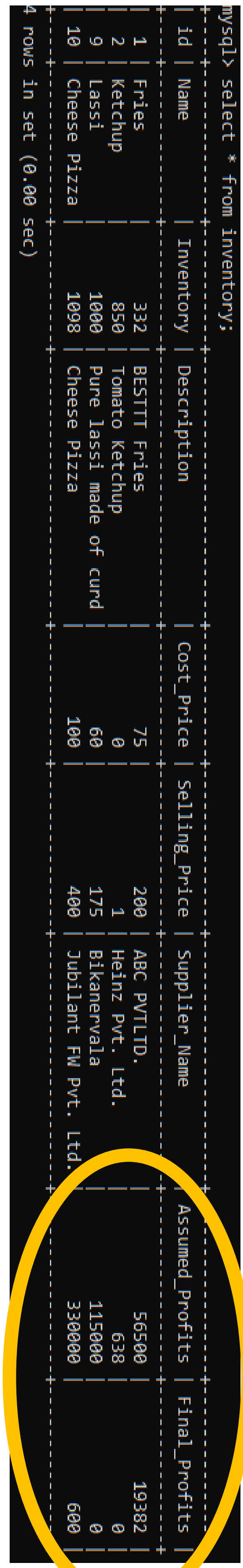
*Highlighted: Changes reflected in table ‘store.inventory’

Output

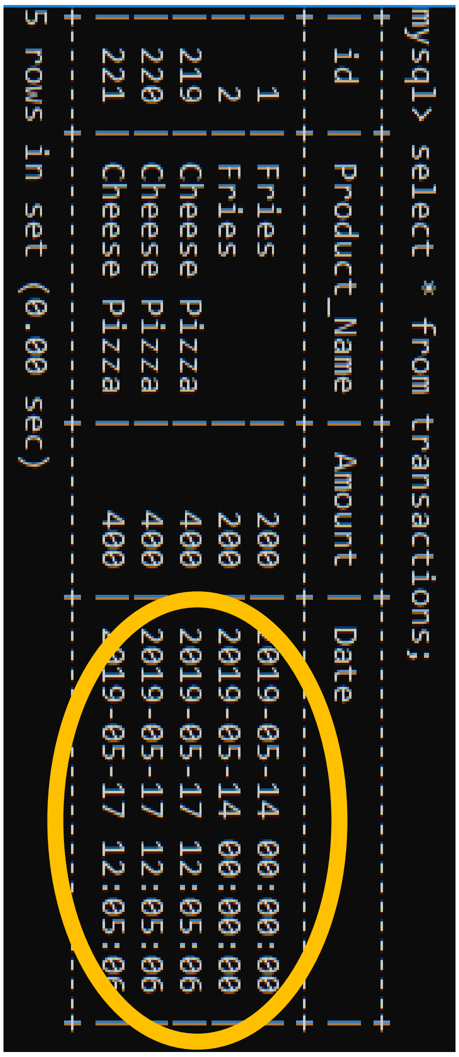
Inventory and Transaction tables are as follows:

Transactions table also shows time and date of item purchase.

Inventory table updates real profits as all items may not be sold and compares them to assumed profits (if all items were sold)

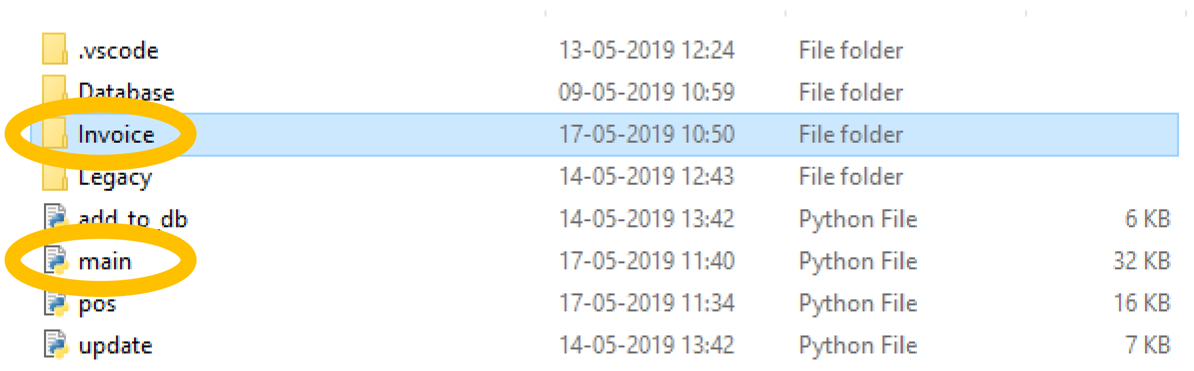


```
mysql> select * from inventory;
+----+--------------+-----------+--------------------------+------------+---------------+-----------------------+-----------------+---------------+
| id | Name         | Inventory | Description              | Cost_Price | Selling_Price | Supplier_Name         | Assumed_Profits | Final_Profits |
+----+--------------+-----------+--------------------------+------------+---------------+-----------------------+-----------------+---------------+
|  1 | Fries        |       332 | BESTTT Fries             |         75 |           200 | ABC PVTLTD.           |           56500 |         19382 |
|  2 | Ketchup      |       850 | Tomato Ketchup           |          0 |             1 | Heinz Pvt. Ltd.       |             638 |             0 |
|  9 | Lassi        |      1000 | Pure lassi made of curd  |         60 |           175 | Bikanervala           |          115000 |             0 |
| 10 | Cheese Pizza |      1098 | Cheese Pizza             |        100 |           400 | Jubilant FW Pvt. Ltd. |          330000 |           600 |
+----+--------------+-----------+--------------------------+------------+---------------+-----------------------+-----------------+---------------+
4 rows in set (0.00 sec)
```



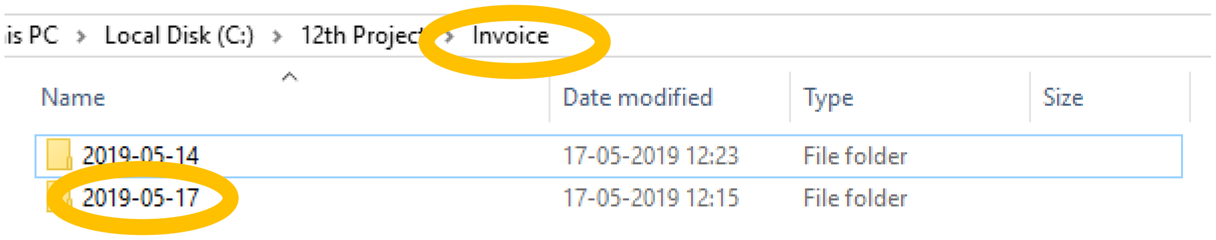
```
mysql> select * from transactions;
+-----+--------------+--------+---------------------+
| id  | Product_Name | Amount | Date                |
+-----+--------------+--------+---------------------+
|   1 | Fries        |    200 | 2019-05-14 00:00:00 |
|   2 | Fries        |    200 | 2019-05-14 00:00:00 |
| 219 | Cheese Pizza |    400 | 2019-05-17 12:05:06 |
| 220 | Cheese Pizza |    400 | 2019-05-17 12:05:06 |
| 221 | Cheese Pizza |    400 | 2019-05-17 12:05:06 |
+-----+--------------+--------+---------------------+
5 rows in set (0.00 sec)
```

Output



.vscode	13-05-2019 12:24	File folder	
Database	09-05-2019 10:59	File folder	
Invoice	17-05-2019 10:50	File folder	
Legacy	14-05-2019 12:43	File folder	
add_to_db	14-05-2019 13:42	Python File	6 KB
main	17-05-2019 11:40	Python File	32 KB
pos	17-05-2019 11:34	Python File	16 KB
update	14-05-2019 13:42	Python File	7 KB

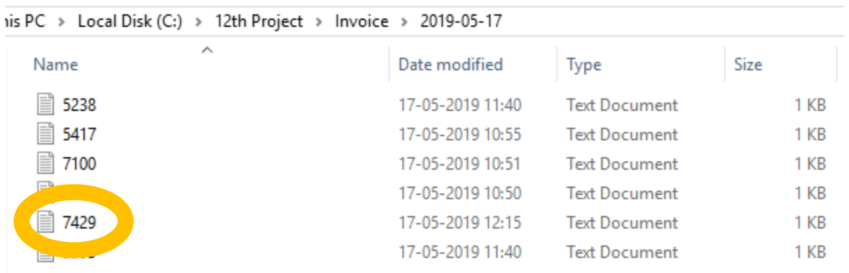
For program *main.py* dynamically invoices are stored in folder at same location *Invoice* using concepts of file creation.



is PC › Local Disk (C:) › 12th Project › Invoice

Name	Date modified	Type	Size
2019-05-14	17-05-2019 12:23	File folder	
2019-05-17	17-05-2019 12:15	File folder	

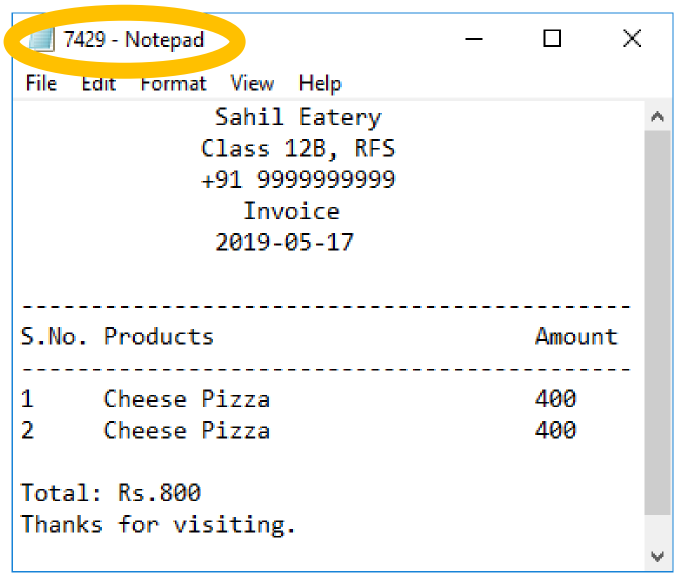
Invoices in *Invoice* folder are saved according to date.



is PC › Local Disk (C:) › 12th Project › Invoice › 2019-05-17

Name	Date modified	Type	Size
5238	17-05-2019 11:40	Text Document	1 KB
5417	17-05-2019 10:55	Text Document	1 KB
7100	17-05-2019 10:51	Text Document	1 KB
[illegible]	17-05-2019 10:50	Text Document	1 KB
7429	17-05-2019 12:15	Text Document	1 KB
[illegible]	17-05-2019 11:40	Text Document	1 KB

A random number (*random.randrange(5000, 10000)*) used to generate random invoices.



```
            Sahil Eatery
           Class 12B, RFS
           +91 9999999999
               Invoice
             2019-05-17

---------------------------------------
S.No. Products                   Amount
---------------------------------------
1     Cheese Pizza               400
2     Cheese Pizza               400

Total: Rs.800
Thanks for visiting.
```

Here is the invoice generated for the sample example used in this output.

Note that even if item name is long, the *Amount* and *S.No.* panels will not shift to the right or left if the name is small.

(*(self.name + ‘   ’)[:14]* used which prevents this)