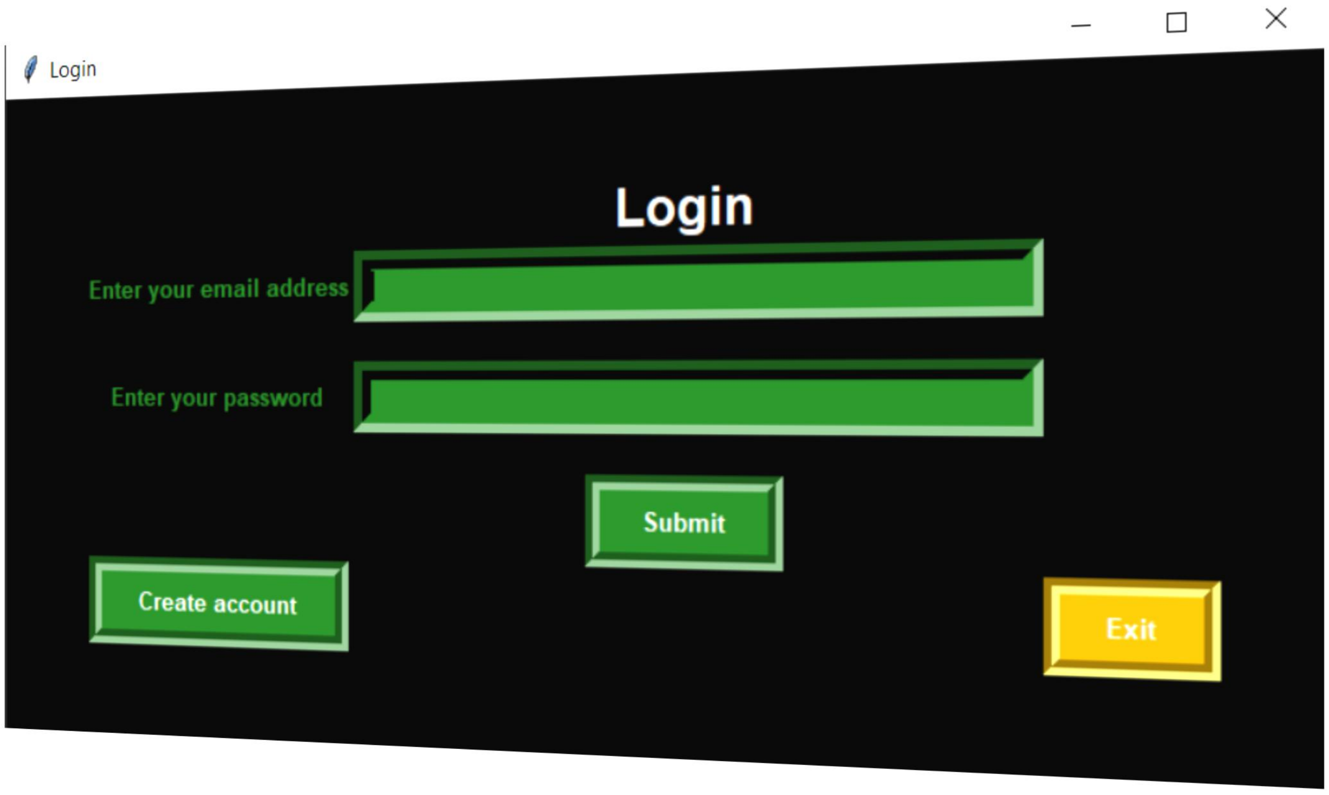
Assignment 4

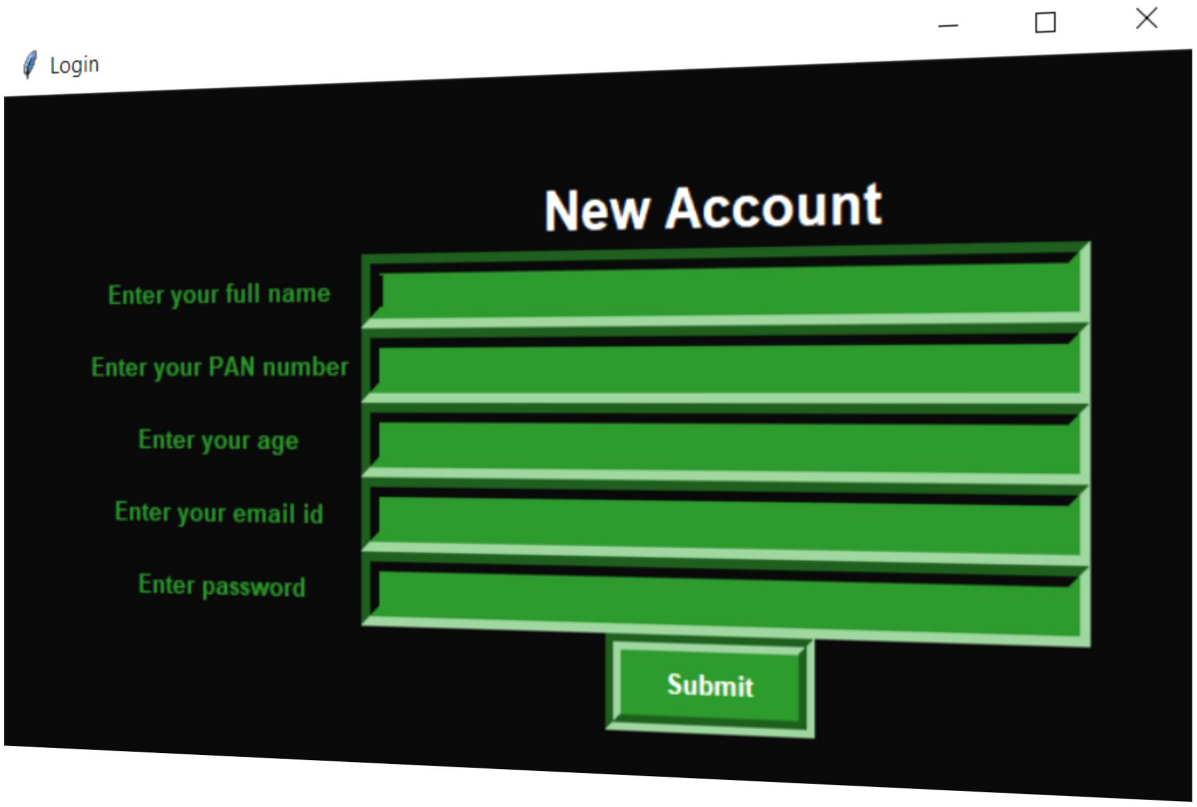
-Vishnu J G(PES1UG19CS574)
Yashwal S Kanchan(PES1UG19CS593)
Tushar Y S(PES1UG19CS545)

Front end details:

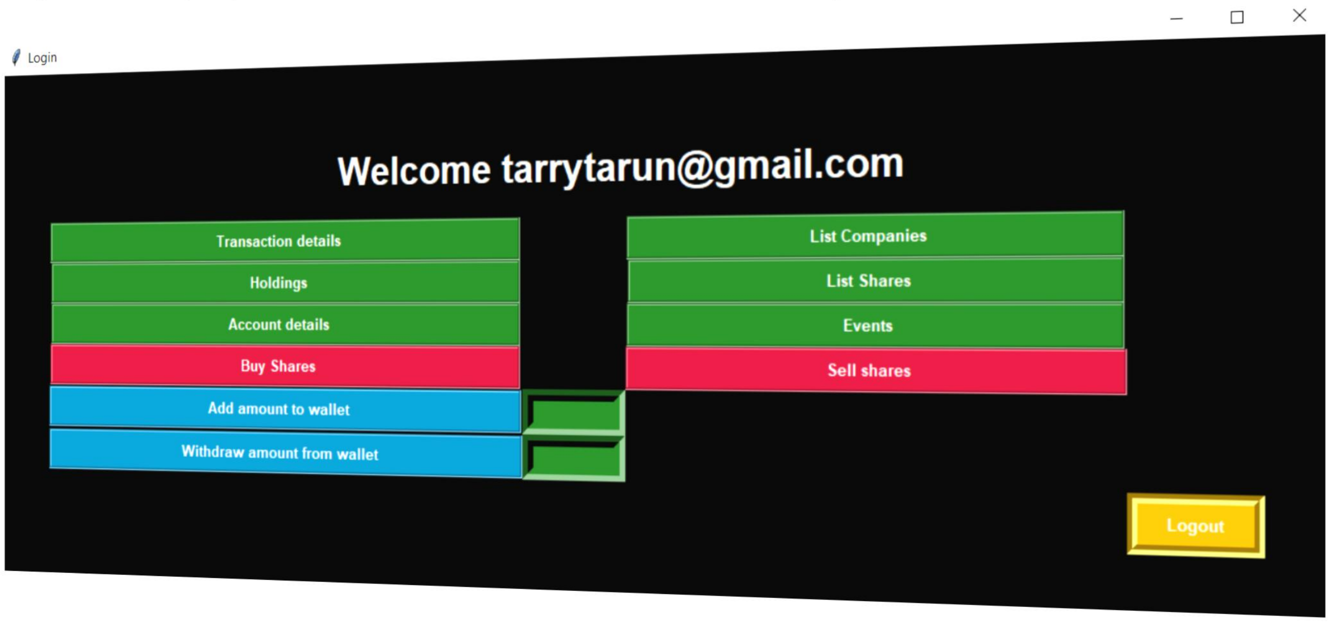
- Front end is written in python using Tkinter module.
- We execute all queries of psql using psycopg2 cursor feature which helps us execute queries and also obtain the output.
- The first page has login info and an option to create a new account(this is an add on to our previously mentions objectives).



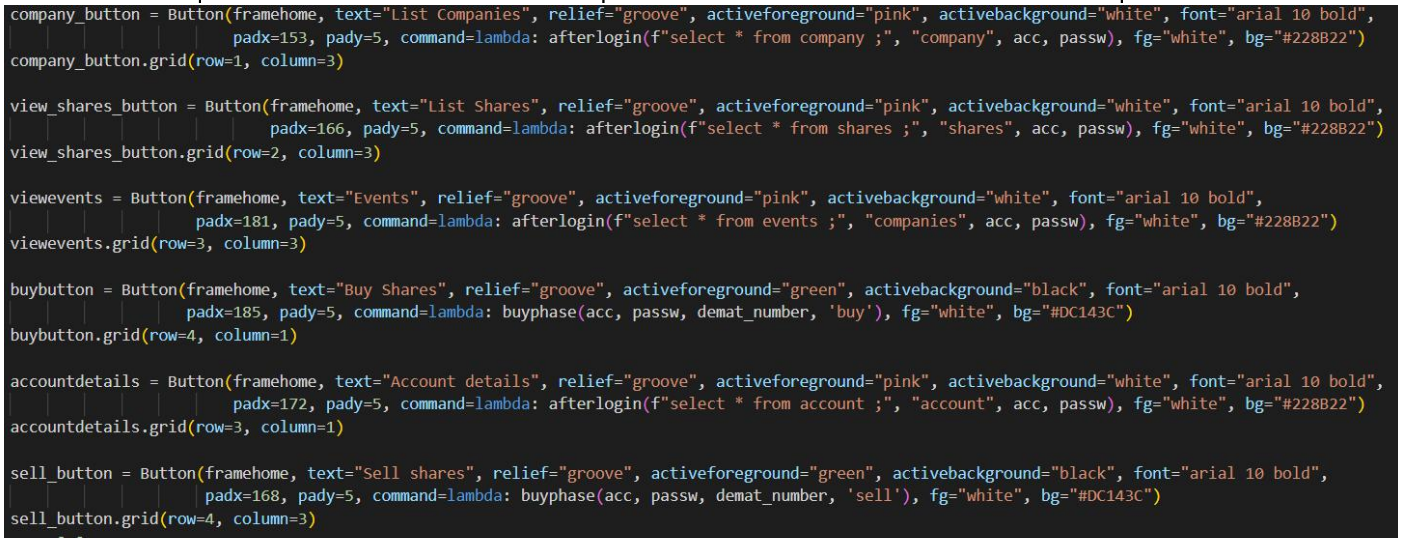
- The create account helps us add new users to the database and grant them different accesses before they can enter the simulation.



- All edge cases have been taken care of. For each error or illegal action, a popup is show specifying the error that occured and the possible solution. The popup is a simple windows popup.
- Upon successful singup or singin, we enter the home page where we are given with a bunch of options such as account details, transactions, etc,. Here is where we can view all the shares and even buy or sell them.

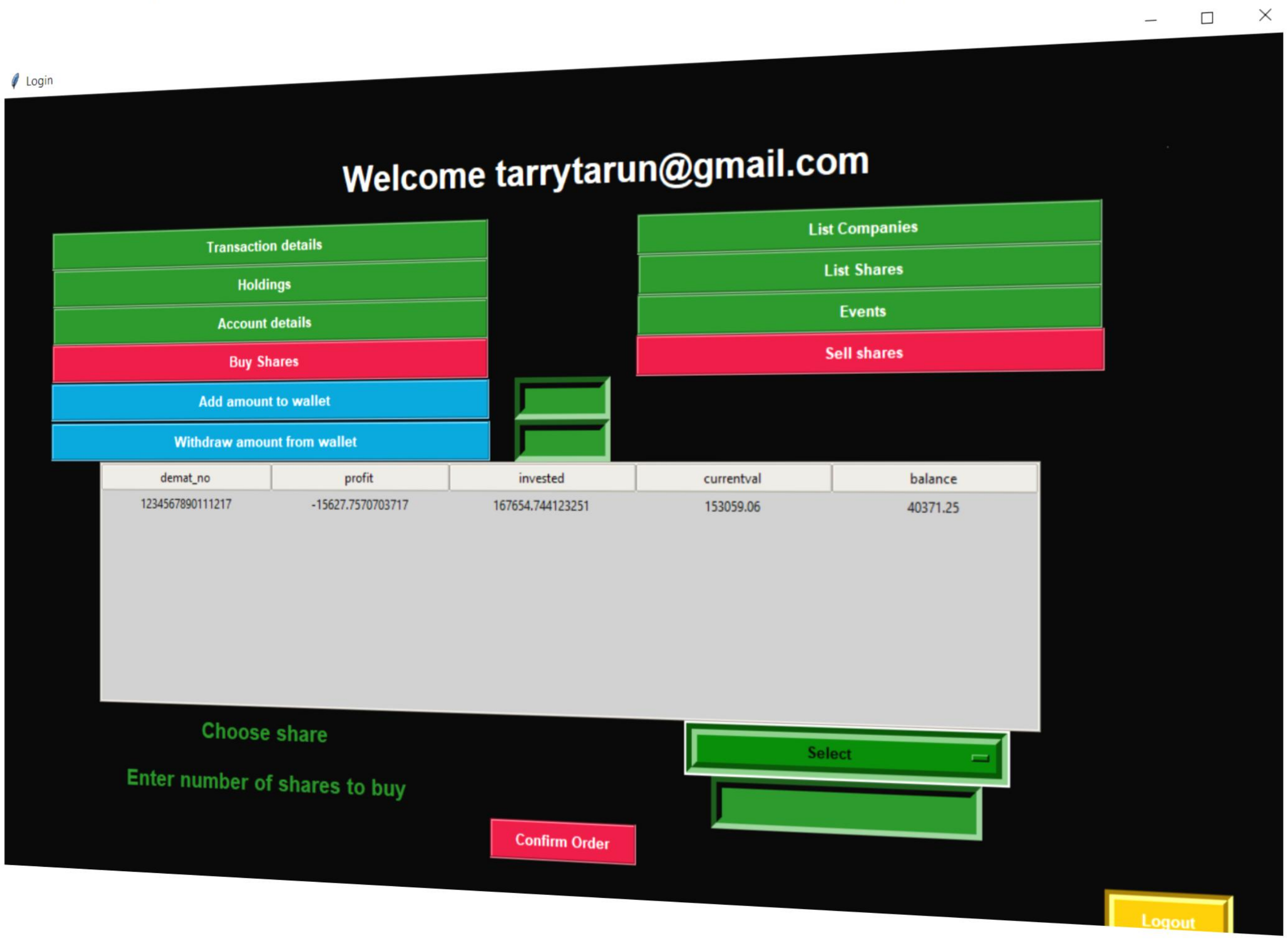


➢ Each button provides different info based on the queries run in the backend. Here are a few examples.



```
company_button = Button(framehome, text="List Companies", relief="groove", activeforeground="pink", activebackground="white", font="arial 10 bold",
                        padx=153, pady=5, command=lambda: afterlogin(f"select * from company ;", "company", acc, passw), fg="white", bg="#228B22")
company_button.grid(row=1, column=3)

view_shares_button = Button(framehome, text="List Shares", relief="groove", activeforeground="pink", activebackground="white", font="arial 10 bold",
                            padx=166, pady=5, command=lambda: afterlogin(f"select * from shares ;", "shares", acc, passw), fg="white", bg="#228B22")
view_shares_button.grid(row=2, column=3)

viewevents = Button(framehome, text="Events", relief="groove", activeforeground="pink", activebackground="white", font="arial 10 bold",
                    padx=181, pady=5, command=lambda: afterlogin(f"select * from events ;", "companies", acc, passw), fg="white", bg="#228B22")
viewevents.grid(row=3, column=3)

buybutton = Button(framehome, text="Buy Shares", relief="groove", activeforeground="green", activebackground="black", font="arial 10 bold",
                   padx=185, pady=5, command=lambda: buyphase(acc, passw, demat_number, 'buy'), fg="white", bg="#DC143C")
buybutton.grid(row=4, column=1)

accountdetails = Button(framehome, text="Account details", relief="groove", activeforeground="pink", activebackground="white", font="arial 10 bold",
                        padx=172, pady=5, command=lambda: afterlogin(f"select * from account ;", "account", acc, passw), fg="white", bg="#228B22")
accountdetails.grid(row=3, column=1)

sell_button = Button(framehome, text="Sell shares", relief="groove", activeforeground="green", activebackground="black", font="arial 10 bold",
                     padx=168, pady=5, command=lambda: buyphase(acc, passw, demat_number, 'sell'), fg="white", bg="#DC143C")
sell_button.grid(row=4, column=3)
```

➢ Row level security makes sure that an user can access information about his account only.



demat_no	profit	invested	currentval	balance
1234567890111217	-15627.7570703717	167654.744123251	153059.06	40371.25

➢ Buy or sell is one that triggers a bunch of queries to update the following table
- Shares table
- A_shares table
- Transactions table
- Accounts table

➢ Deposit or withdraw to or from balance also triggers update on accounts table

The changes to schema:

- The a_shares table, which has the holdings of an user has been updated. It now has a column average which is a derived attribute which tells at what average price was the share bought.
- The a_shares table also has a profit column to specify the profit per stock. This way users can see how much profit they have made per company stock.
- Few additional queries include, update of a_shares where we delete column with 0 shares. Updating the average values, updating the share prices each time someone signs in, updating balance and profit when a transaction such as buy or sell is performed.
- One major update is change of variables for transactions table from 0 to 1 for buy and 1 to -1 for sell. This is major requirement cause this helps us calculate the average price, which is calculated as sum((buy or sell) * quant).

Data migration features:

- The database is easy to migrate, since almost all setup is from the insert.sql and create.sql which is also optional. Which is why we can do the complete database creating operation from python, I.e, the front end.
- Most of the features implemented in the frontend can do the basic setup and running of the database.
- Migration to a different database other than postgres, can cause problems as the queries are hard coded in psql language.

Dependencies installation:

Apart from Postgres, the python modules include :

- Pynse: used to stream live data from stock market.
- Tkinter: complete front end is designed in tkinter and it features.
- Datetime : to create psql acceptable datetime values
- Psycopg2 : To access the database and execute the queries.

Contributions:

Vishnu J G : Font end and back end integration.(12 hrs)
Yashwal S Kanchan : Query execution and transactions.(12 hrs)
Tushar Y S : Trigger and Assertions.(12 hrs)