**CSC 521**

**Advanced OO programming with Java and Python**



****

Project Supervisor - Professor Da Qi Ren

**Team Members:**

| **Name** | **Student ID** |
| --- | --- |
| Savita Khadse | 27565948 |
| Samee Peerzade | *59638609* |
| Sagar Tandel | *17857240* |

**Acknowledgement**

We take this opportunity to thank Professor Da Qi Ren Kim, International Technological University, Santa Clara for the valuable support and guidance extended to us, which helped in the successful completion of the project.

**TABLE OF CONTENTS**

1. Abstract
2. System Requirements
3. Introduction
4. Technical Structure
5. Use cases diagram
6. Class Diagram
7. Activity Diagram
8. Code
9. Conclusion
10. Screenshots of game
11. Steps to play game

**Abstract**

“Wordle” is a game where the player tries to guess a secret word based on the hints that will be provided to the player in Wordle. In Wordle, users have to guess the word that has five letters over six attempts by using a color-hint strategy and **Hint** label provided to show the first letter of the secret word which will only display if you hover cursor on the label or you can choose to play game based on color hint.

Once the player guesses the word, the input letters will be reflected as yellow,gray or green tiles. For example, a **green tile** means that the player predicted the correct letter and placement of a letter of the final word. A **yellow tile** means that the letter is present in the final word, but the letter is in the wrong position. The **gray tile** means a letter does not exist in the correct word.

If the user fails to guess the correct word, the secret word will be displayed after the 6th attempt and then you can click to play the next word and start the game with a fresh new word.

**System Requirements:**

We focused on the following set of requirements while designing the Wordle game:

1. In order to play the game the new user has to **register** for the game
2. After a successful registration, the user will be able to **login** to the game using registered credentials
3. In the main game page, the user will find five boxes and six rows to try and guess the word. Six rows are for six attempts to guess the final word and five boxes represent a five letter word.
4. In every attempt the user needs to try and guess the word, if wrong he/she should get a message to display the guess is incorrect and try again..
5. If the user guesses the word correctly before six and last attempt, a message should popup to display he guessed the word and option to go to next level or logout the game.
6. Once the user select play again, a game panel should clear out old data and start with first attempt for new word
7. For additional help or to reduce difficulty level, a hint is there which tells the user the first letter of a game.

**Introduction**

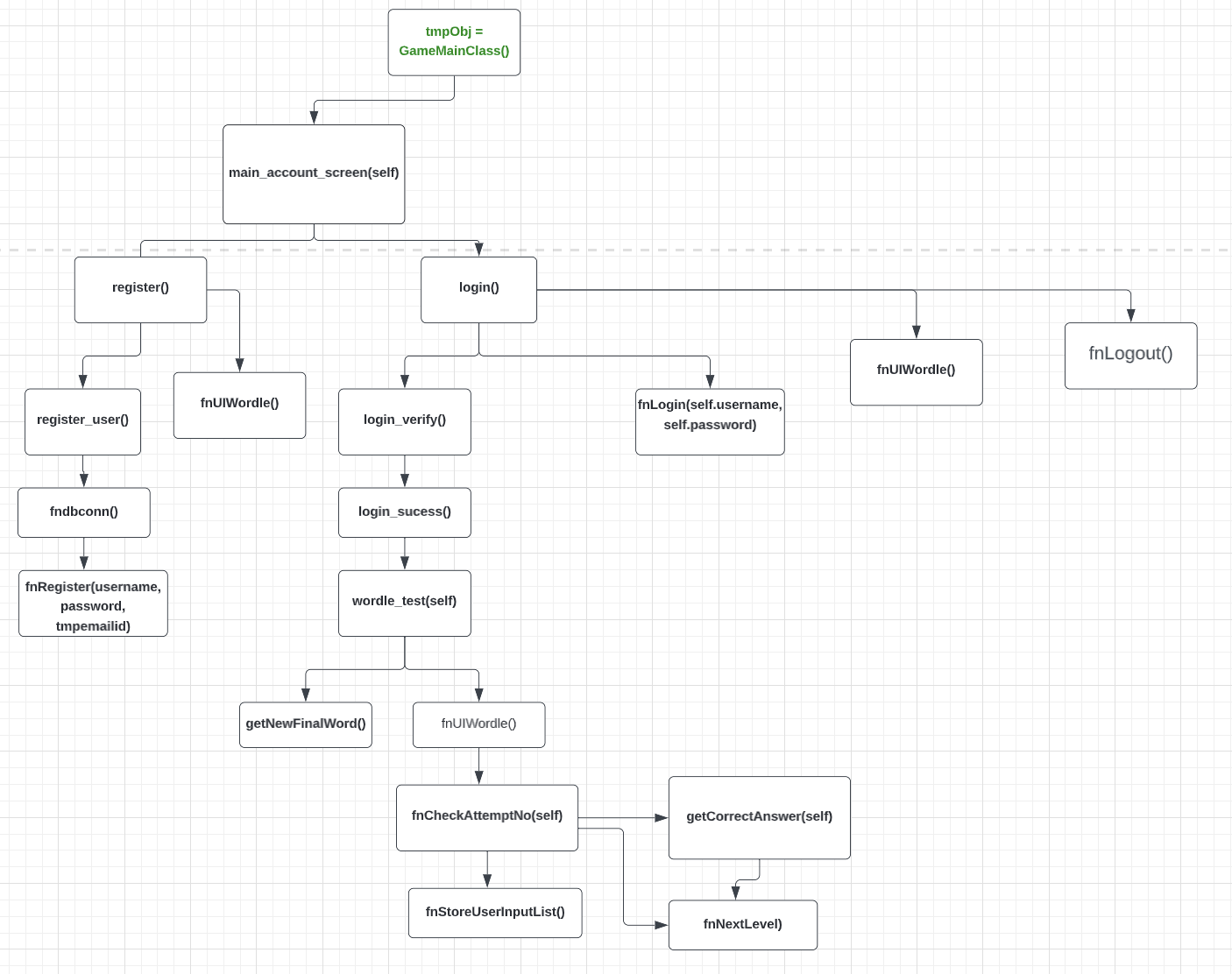
The Wordle game is created with object oriented programming (OOP) concepts. It includes two classes, multiple functions and imported standard packages and libraries such as tkinter, random, etc.

We have also created a python to MySql database connection to save users data (username, password and email id) in ‘users’ table to verify user login credentials at the time of login request. Programming languages like Python need a special driver before they can speak to a database from a specific vendor. A database driver is a piece of software that allows an application to connect and interact with a database system. So we have installed MySQL connector python to establish connection between users table and python code.

Login pages are important for the development of any kind of mobile or web application. This page is most essential for user authentication purposes.

**Technical Structure**

This game is designed using six classes and 19 functions.



In order to play the game the new user has to register for the game. When the first game screen pops up, the user will click register to enter his/her details in the Account Login Window function **register\_user() from RegisterClass** gets called. In that Register Window the user has to give their details about username, password, and email. After clicking the **Register** the information will be saved in the database’s ‘users’ table by using **fnRegister() from mysqlCon class** and then a message box comes up saying “Registration done successfully!!”.

After a successful registration, the user will be able to login to the game using registered credentials through the login page window where after clicking submit **login()** function will be executed. The details will get checked in the database by using the **login\_verify() function** **from mysqlCon class**. If the credentials are successful, the function **login\_sucess()** and **wordle\_test()** gets executed and a pop-up appears saying ‘welcome to game!’. If not, another function called **password\_not\_recognised()** will get invoked. That def **password\_not\_recognised()** function will display a pop-up ‘Invalid password’.

After a successful login, the main screen of a game appears through the function **wordle\_test() from GameMainClass class** in front of the user**.** In the main game page, the user will find six rows and five boxes in each row. Six rows are for six attempts to guess the final word and five boxes in each row represent the final word as a five letter word.

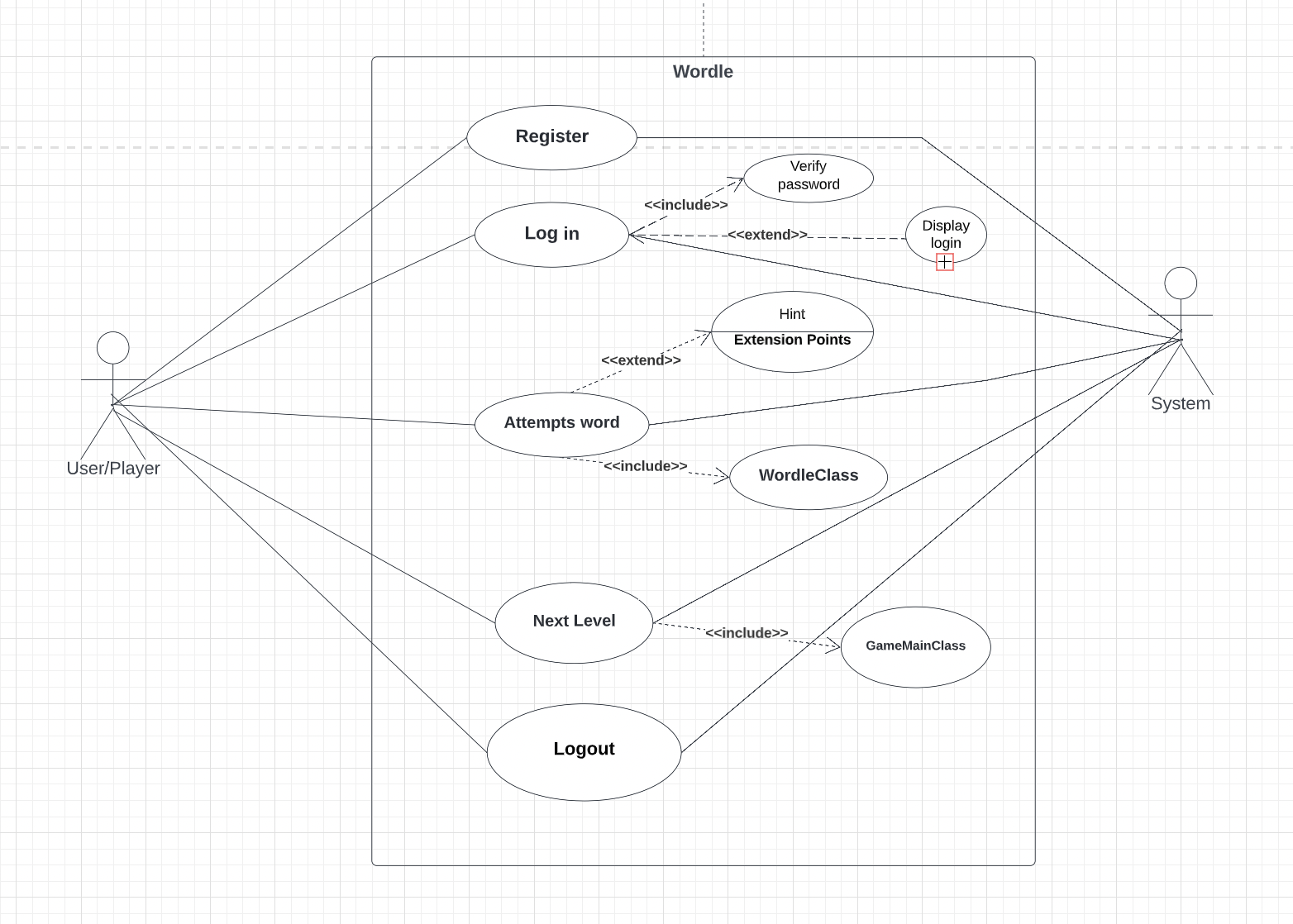
In every attempt the user needs to try and guess the final word, after the sixth attempt the user will lose the game and the final answer will be displayed. To check a user's answer with the final word function **fnStoreUserInputList()** and **fnCheckAttemptNo()** gets executed and these methods belong to the **UserAttemptClass class.** The for loop will iterate for each letter of the user's answer and check if the letter and position of the letter is correct and flags will be set and **Greenlist, yellow list** and **greylist** will be appended accordingly.

The function of the **fnCheckAttemptNo()** is to check the greenlist, yellow list and greylist values and color the box as per the number appearing in those lists. For example, for Attempt\_No#1, greenlist has [1,3,5] then the first row’s box no 1 , 3 and 5 will become green. If yellowlist has [2] in it then box no 2 of that row will become yellow and if greylist has [4] then 4th box of that will become gray. This will be checked for every row till the sixth row.

Once the condition **if len(greenList) == 5: is** met all boxes of that rows become green and the user will get the “Congratulations! Your answer is correct.” message. Otherwise, the user will get “**Incorrect answer! Try again**” message. After all six failed attempts the message will pop up saying “**Incorrect answer! No attempts remain.**” and the final word will be displayed through the function **getCorrectAnswer() from WordleClass.** The player can choose to play another word by clicking the next level button where the function **fnNextLevel() and wordle\_test(again)** will be executed.

Finally, if the user wants to quit the game, he/ she can click the logout button and a function called **fnLogout() which belongs to UserAttemptClass** will be invoked which will close the game window. There is another feature in this game that gives a hint to the user, that feature will display the first letter of the final word.

**Use Case Diagram:**

****

**Class Diagrams:**

1. Class mysqlCon
2. Class WordleClass
3. Class RegisterClass
4. Class LoginClass
5. Class GameMainClass
6. Class UserAttemptClass(GameMainClass)

Let’s start studying in detail about our class diagram and this OOPS concept we used.

We have tried to avoid duplication by using the Object-Oriented Programming concept mostly by focusing on Inheritance. Interesting part of the OOPS concept and Inheritance is, you don’t even need to inherit multiple times.

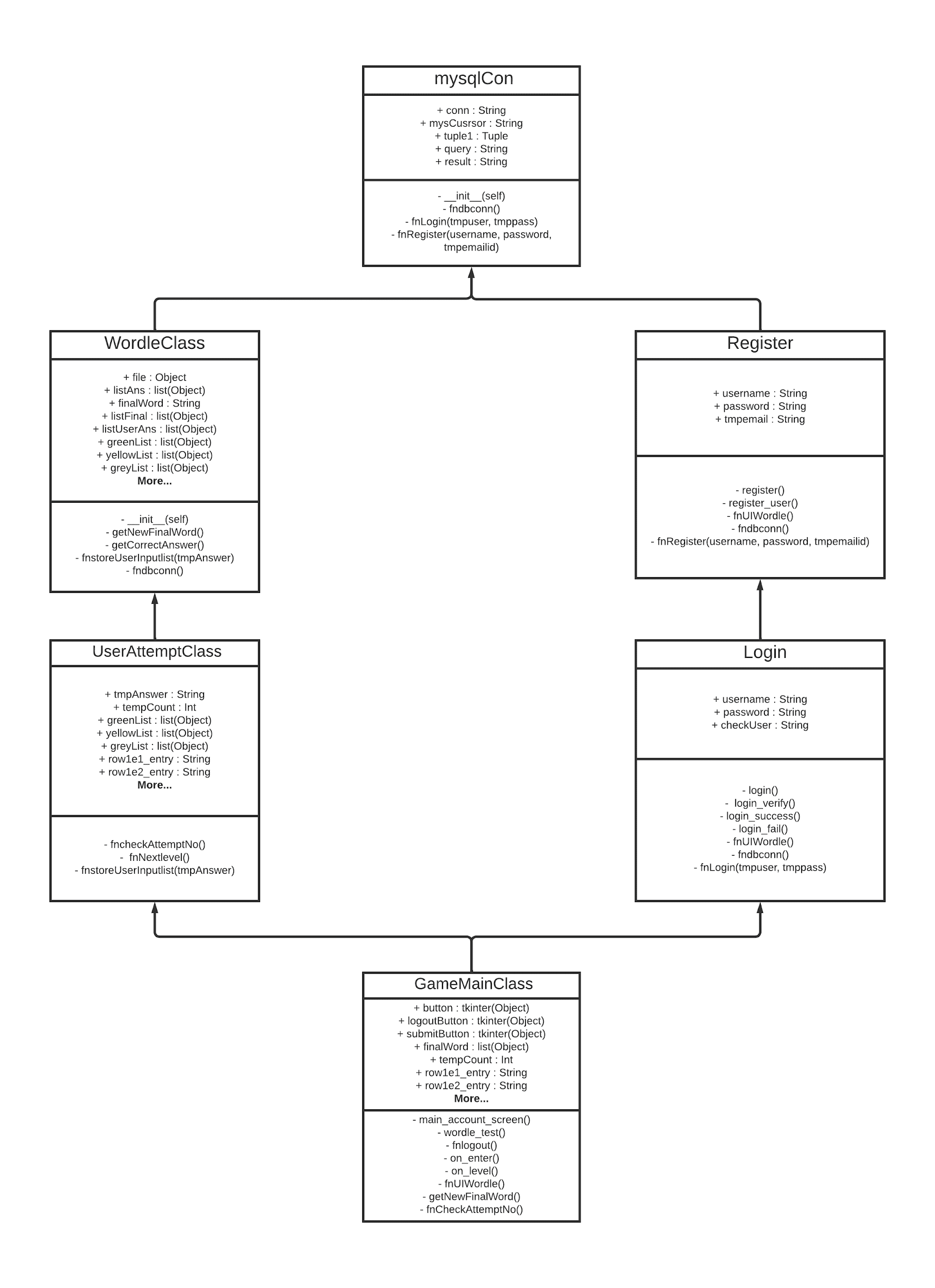
First, we have created a class called “mysqlCon” where we have achieved python to the mysql Connection to register our user and then to authenticate them if they already registered before by using methods called “fndbConn()”, “fnRegister()” and “fnLogin()”. These three functions have assigned specific tasks that will be performed on calling these functions. But to access these functions, you need to Inherit class “mysqlCon” as we did in our next two classes called “Register” and “Login”.

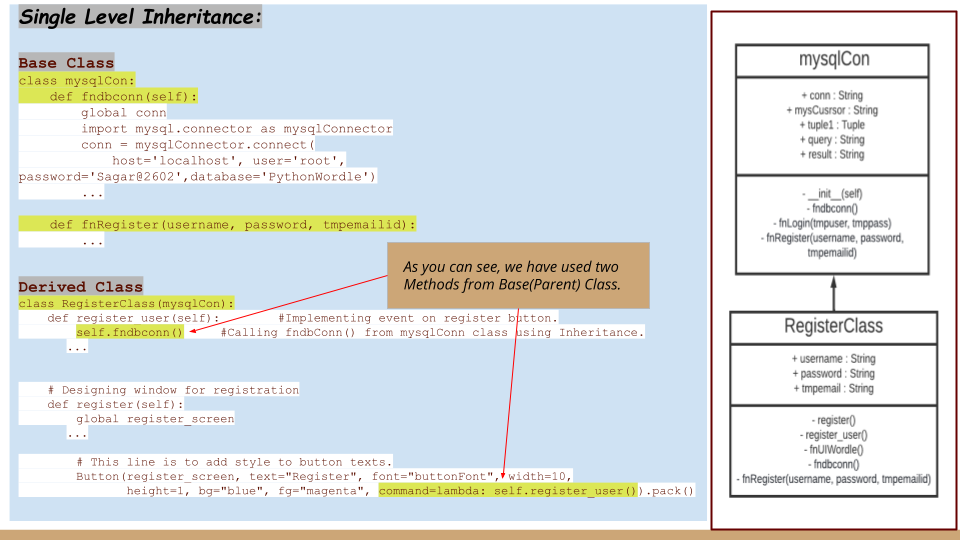
As “RegisterClass” is already inheriting mySqlCon, “LoginClass'' doesn't need to inherit “mysqlCon” class instead “LoginClass” just need to inherit “RegisterClass” and “LoginClass” now have access of “mysqlConn” class as well. This is **Multilevel inheritance.**

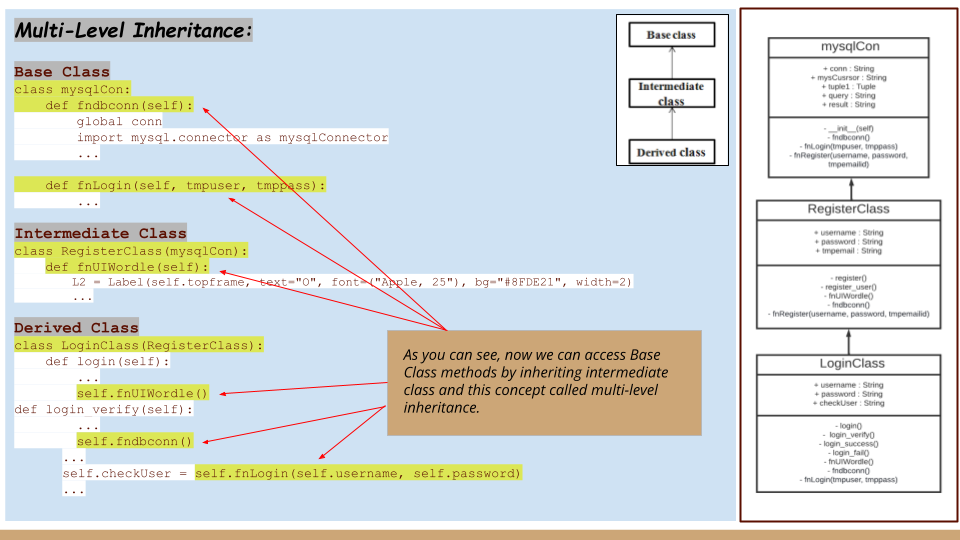
There are other important classes which are for our actual game to work and they also use the Multilevel Inheritance concept. If you check, “WordleClass” is also inheriting mySqlCon and “UserAttemptClass” inheriting WordleClass. So all the functions from parent Classes are accessible to respective child classes by this Multilevel Inheritance concept.

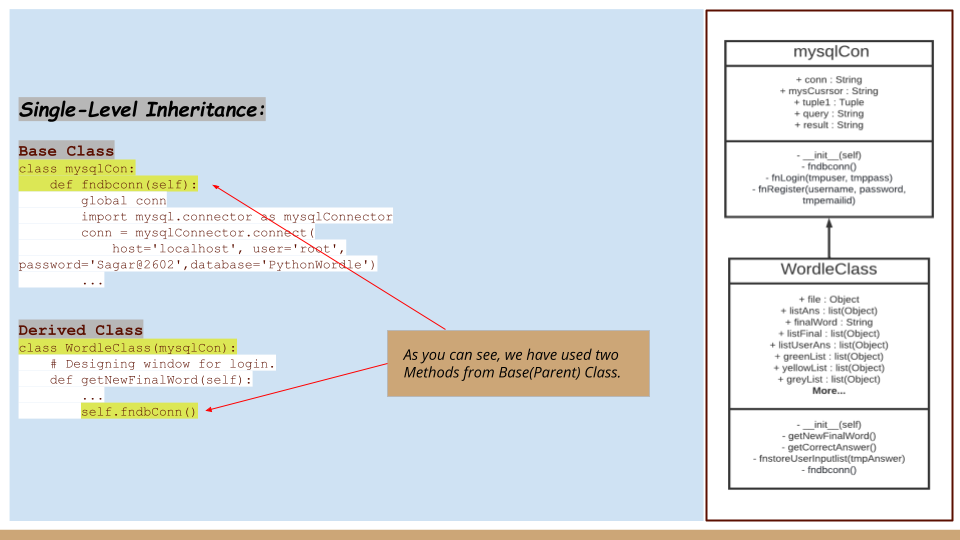
We have also used the Multiple level inheritance to access functions from “UserAttemptClass” and “LoginClass” in the “GameMainClass”.

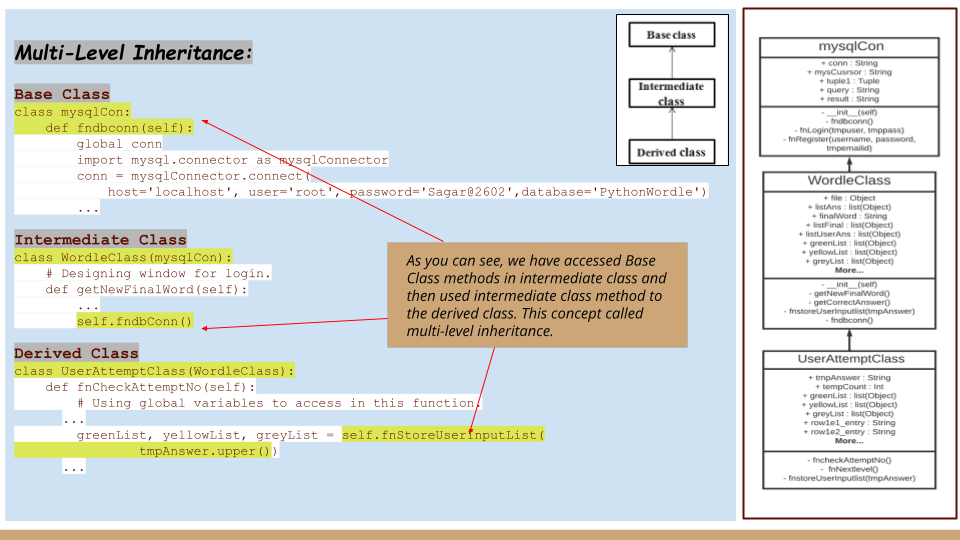
So as you can see we have tried Multilevel, Multiple inheritance in our program to avoid maximum duplication and structured source code in such a way to act the game properly to make it work in every scenario.

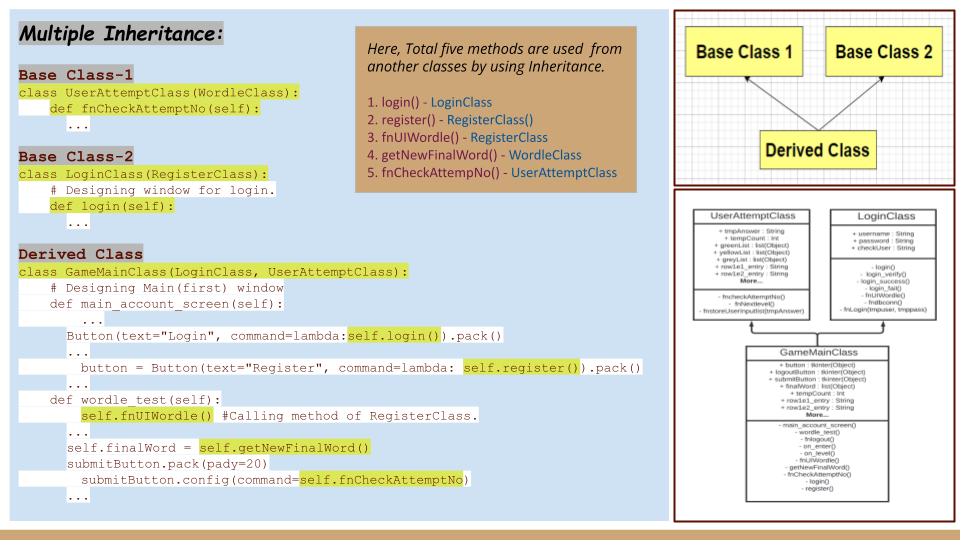


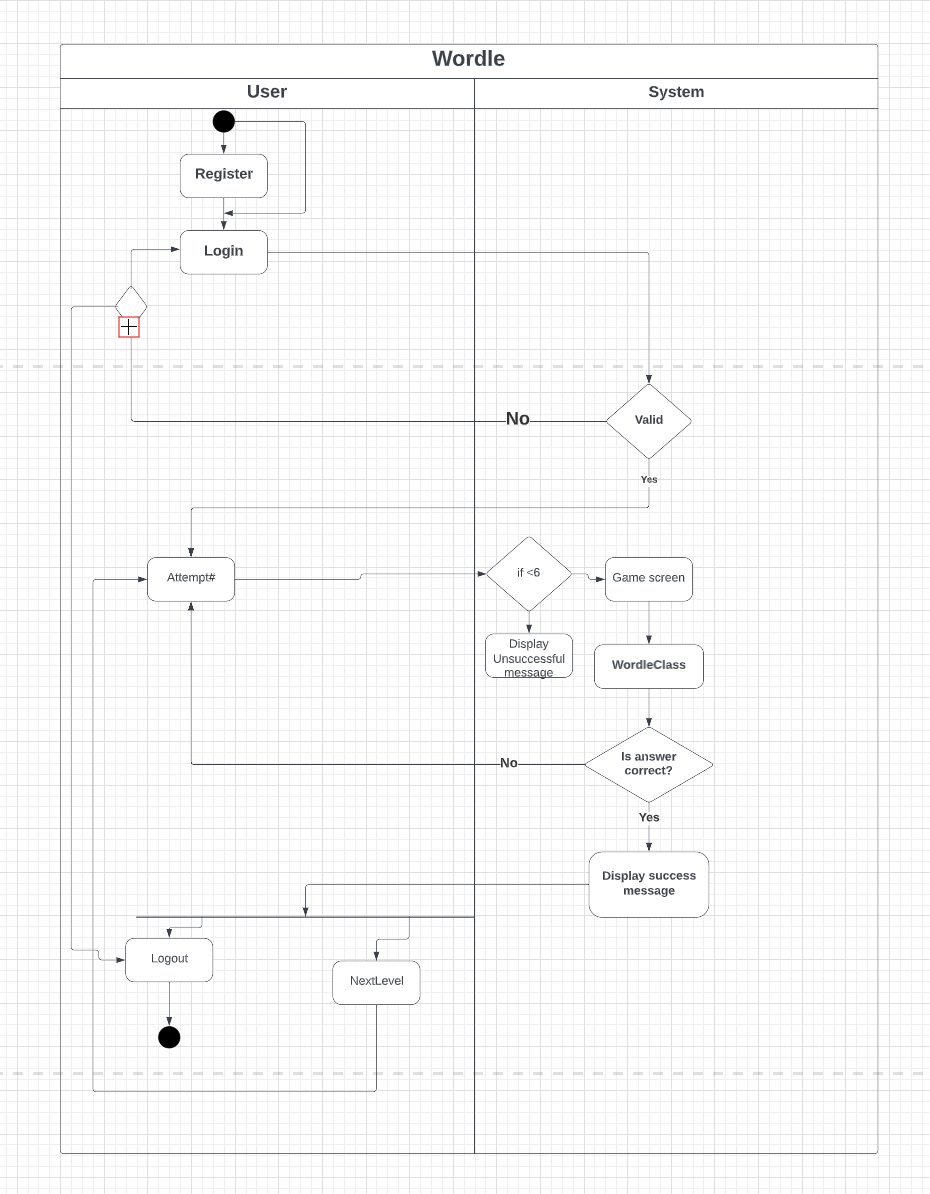
****

****

****

****

****

**ActivityDiagram:**

**Graphical User Interface(GUI) design:**

Python GUI is a graphical user interface that allows the user to interact with the operating system and other programs using graphical elements such as icons, buttons,and dialog boxes.

In Python, you can use the tkinter module to create simple GUI programs. Python does not have GUI programming features built into the language itself. However, it comes with a module named tkinter that allows you to create simple GUI programs.

A GUI program presents a window with various graphical widgets with which the user can interact or view. We have used some Label widgets to create a line of text or image, for example, username, password and email. Text widgets are also used to enter the texts like username, password and email by the user. We have also used the Button widget for clicking the buttons. We have used Frame widget, that is a container that can hold other widgets. Each widget has different properties like, background color, foreground color, width, height, font, internal and external padding, anchor etc.

**Source Code:**

**1. Database connection:**

**Tkinter\_dbconnection\_file.py**

from logging import exception

#from sqlite3 import Cursor

from mysql.connector.cursor import MySQLCursor

class mysqlCon:

conn = ""

def \_\_init\_\_(self): # self represents the instance of the class. Usage of "self" in class to access the methods and attributes

print("This is as an initializer method of class mysqlCon ...")

def fndbconn(self):

global conn

# Importing the connector class from MySql.

import mysql.connector as mysqlConnector

conn = mysqlConnector.connect(

host='localhost', user='root', password='rootroot1234', database='PythonWordle') # We access the connect method through the connector class, which we already import into our program. Now, we are passing our connection parameters to the connect method. The user name and password will be different according to your installation process.

if conn:

return 1

else:

return 0

def fnLogin(self, tmpuser, tmppass):

global conn

# Importing cursor method from the established connection (conn) object and createing the cursor object mycursor to use in code.

mycursor = conn.cursor()

print(f"Entered Username = {tmpuser}")

print(f"Entered Password = {tmppass}")

tuple1 = (tmpuser, tmppass)

query = """select \* from users where username=%s and password=%s"""

# The execute () method helps us to execute the query and return records according to the query.

result = mycursor.execute(query, tuple1)

# Returns the all or remaining rows from the result set.

result = mycursor.fetchone()

print(f"Result = {result}")

return result

def fnRegister(username, password, tmpemailid):

mycursor = conn.cursor()

print(username, password, tmpemailid)

mycursor.execute(

f"insert into users(username,password,email\_id) values('{username}','{password}','{tmpemailid}')")

conn.commit()

conn.close()

**Tkinter\_login\_and\_register\_five\_letter.py**

#import modules

from math import radians

from tkinter\_dbconnection\_file import mysqlCon

from cgitb import text

from tkinter import \*

from tkinter import font

from tkinter import END, Label, messagebox

# This module provides a portable way of using operating system dependent functionality.

import os

from tkinter import messagebox

from operator import length\_hint

import random

# Global Variable Declaration.

global tmpAnswer

global tmpRandomChoice

tempCount = 1 # This variable to call word to check as per the attempts.

# importing database class from file name called "tkinter\_dbconeection\_file.py"

class WordleClass(mysqlCon):

def \_\_init\_\_(self):

print("Hello this is wordleclass...")

self.fndbconn()

def getNewFinalWord(self):

global tmpRandomChoice

# This code is to get final word list from file.

self.listAns = []

file = open(

"/Users/savitakhadse/Desktop/Summer 2022/Advanced OO programming with Java and Python/Project/New version/wordle.txt", "r")

self.listAns = file.read().split('\n')

file.close()

# Generating random number's word for today's game.

tmpRandomChoice = random.randrange(0, 25)

# This to store random generated word for Game.

i = 1

self.finalWord = self.listAns[tmpRandomChoice]

print(self.finalWord)

self.listFinal = []

self.listFinal.clear()

for i in range(5):

self.listFinal.append(self.finalWord[i])

print(self.listFinal)

return self.finalWord

def getCorrectAnswer(self):

print(f"Correct Answer = {self.finalWord}")

return self.finalWord

def fnStoreUserInputList(self, tmpAnswer):

# Here, starting the storing user input in list from here.

self.listUserAns = []

self.listUserAns.clear()

j = 1

for j in range(5):

self.listUserAns.append(tmpAnswer[j])

print(self.listUserAns)

# Matching User and Final Answer.

# This list to be used to make TKINTER UI , green-yellow-grey as per the answer.

global greenList

global yellowList

global greyList

greenList = []

yellowList = []

greyList = []

# -------------End ------------.

self.countC = 0

self.countI = 0

k = 0

l = 0

# LOOP TO CHECK USERS WORD LETTER WITH COMPUTER GENERATED WORD LETTER

for k in range(5):

print(f"\nChecking result for the #{k+1} letter.")

flag1 = 0

flag2 = 0

flag3 = 0

for l in range(5):

user = self.listUserAns[k]

computer = self.listFinal[l]

# user==compter is used for letter matching, k==l is used for position matching.

if user == computer and k == l:

flag1 = 1

elif user == computer and k != l:

flag2 = 2

elif user != computer:

flag3 = 3

else:

pass

# Check and add answer in respective lists.

if flag1 == 1:

# print(f"Position of letter #{user} is correct.")

print(f"\033[1;32;40m {user} \n")

self.countC += 1

greenList.append(k+1)

elif flag2 == 2 and flag3 == 3:

# print(f"This #{user} letter is exist but position is not correct.")

print(f"\033[1;33;40m {user} \n")

self.countI += 1

yellowList.append(k+1)

elif flag3 == 3:

# print(f"This #{user} letter is not exist.")

print(f"\033[1;37;40m {user} \n")

self.countI += 1

greyList.append(k+1)

else:

pass

# Calling function to check final answer and user limits.

return greenList, yellowList, greyList

class RegisterClass(mysqlCon): # This class is for Registration page.

def register\_user(self): # Implementing event on register button.

# Calling fndbConn() from mysqlConn class using Inheritance.

self.fndbconn()

self.username\_info = self.username.get()

self.password\_info = self.password.get()

self.email\_info = self.tmpemail.get()

self.username\_entry.delete(0, END)

self.password\_entry.delete(0, END)

self.email\_entry.delete(0, END)

checkUser = mysqlCon.fnRegister(

self.username\_info, self.password\_info, self.email\_info)

messagebox.showinfo(

"Success!", "Welcome! Registration Done Suceessfully..")

register\_screen.destroy()

def fnUIWordle(self):

L1 = Label(self.topframe, text="W", font=(

"Apple, 25"), bg="#8FDE21", width=2)

L2 = Label(self.topframe, text="O", font=(

"Apple, 25"), bg="#8FDE21", width=2)

L3 = Label(self.topframe, text="R", font=(

"Apple, 25"), bg="#8FDE21", width=2)

L4 = Label(self.topframe, text="D", font=(

"Apple, 25"), bg="#8FDE21", width=2)

L5 = Label(self.topframe, text="L", font=(

"Apple, 25"), bg="#8FDE21", width=2)

L6 = Label(self.topframe, text="E", font=(

"Apple, 25"), bg="#8FDE21", width=2)

L6.pack(side=RIGHT, padx=3, pady=10)

L5.pack(side=RIGHT, padx=3)

L4.pack(side=RIGHT, padx=3)

L3.pack(side=RIGHT, padx=3)

L2.pack(side=RIGHT, padx=3)

L1.pack(side=RIGHT, padx=3)

# Designing window for registration

def register(self):

global register\_screen

register\_screen = Toplevel(main\_screen)

register\_screen.title("Register")

register\_screen.geometry("350x500")

# Added for Wordle....

iframe = Frame(register\_screen)

iframe.pack()

self.topframe = Frame(register\_screen)

self.topframe.pack()

mframe = Frame(register\_screen)

mframe.pack()

L1 = Label(iframe, font=("Apple, 10"))

L1.pack()

# Calling method of RegisterClass to create UI of "WORDLE" word so no need to repeat this task again and again.

self.fnUIWordle()

L1 = Label(mframe, font=("Apple, 10"))

L1.pack()

# End of wrodle UI....

self.username = StringVar()

self.password = StringVar()

self.tmpemail = StringVar()

Label(register\_screen, text="Please enter details below", bg="blue",

fg="white", width="300", height="2", font=("Calibri", 13)).pack()

Label(register\_screen, text="").pack()

username\_lable = Label(register\_screen, text="Username \* ")

username\_lable.pack()

self.username\_entry = Entry(

register\_screen, textvariable=self.username)

self.username\_entry.pack()

password\_lable = Label(register\_screen, text="Password \* ")

password\_lable.pack()

self.password\_entry = Entry(

register\_screen, textvariable=self.password, show='\*')

self.password\_entry.pack()

email\_lable = Label(register\_screen, text="Email \* ")

email\_lable.pack()

self.email\_entry = Entry(register\_screen, textvariable=self.tmpemail)

self.email\_entry.pack()

Label(register\_screen, text="").pack()

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

Button(register\_screen, text="Register", font="buttonFont", width=10,

height=1, bg="blue", fg="magenta", command=lambda: self.register\_user()).pack()

# This class is for the Login Page.

class LoginClass(RegisterClass):

# Designing window for login.

def login(self):

global login\_screen

login\_screen = Toplevel(main\_screen)

login\_screen.title("Login")

login\_screen.geometry("350x500")

# Added for Wordle....

iframe = Frame(login\_screen)

iframe.pack()

self.topframe = Frame(login\_screen)

self.topframe.pack()

mframe = Frame(login\_screen)

mframe.pack()

L1 = Label(iframe, font=("Apple, 10"))

L1.pack()

# Calling method of RegisterClass to create UI of "WORDLE" word so no need to repeat this task again and again.

self.fnUIWordle()

L1 = Label(mframe, font=("Apple, 10"))

L1.pack()

# End of wrodle UI....

Label(login\_screen, text="Please enter details below to login", bg="blue",

fg="white", width="300", height="2", font=("Calibri", 13)).pack()

Label(login\_screen, text="").pack()

self.username\_verify = StringVar()

self.password\_verify = StringVar()

Label(login\_screen, text="Username \* ").pack()

self.username\_login\_entry = Entry(

login\_screen, textvariable=self.username\_verify)

self.username\_login\_entry.pack()

Label(login\_screen, text="").pack()

Label(login\_screen, text="Password \* ").pack()

self.password\_login\_entry = Entry(

login\_screen, textvariable=self.password\_verify, show='\*')

self.password\_login\_entry.pack()

Label(login\_screen, text="").pack()

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

Button(login\_screen, font="buttonFont", text="Login", bg="blue",

fg="magenta", width=10, height=1, command=lambda: self.login\_verify()).pack()

# Implementing event on login button

def login\_verify(self):

self.username = self.username\_verify.get()

self.password = self.password\_verify.get()

self.username\_login\_entry.delete(0, END)

self.password\_login\_entry.delete(0, END)

self.fndbconn()

# checking here if the details are matching with mysqlconntable

self.checkUser = self.fnLogin(self.username, self.password)

print(self.checkUser)

if (len(self.checkUser) > 0):

Label(login\_screen, text="Login Success",

fg="green", font=("calibri", 11)).pack()

login\_screen.destroy()

# Calling Wordle Game Main UI.

self.login\_sucess()

else:

self.password\_not\_recognised()

# Designing popup for login success

def login\_sucess(self):

tmp = "Login Successfully!\nWelcome '" + self.username + "'!"

messagebox.showinfo("Success!", tmp)

main\_screen.destroy()

GameMainClass.wordle\_test(self)

# Designing popup for login invalid password

def login\_fail(self):

messagebox.showinfo("Fail!", "Unauthorizes Access. Try Again!")

class UserAttemptClass(WordleClass):

def fnCheckAttemptNo(self):

# Using global variables to access in this function.

global tmpLabel

global wordle\_screen

# End

# Checking attempt and then to check the user answer with final answer.

print(f"Attempt# = {self.tempCount}")

# --- For First attempt ----

if self.tempCount == 1:

tmpAnswer = row1e1\_entry.get() + row1e2\_entry.get() + row1e3\_entry.get() + \

row1e4\_entry.get() + row1e5\_entry.get()

row1e1\_entry.config(state="disabled")

row1e2\_entry.config(state="disabled")

row1e3\_entry.config(state="disabled")

row1e4\_entry.config(state="disabled")

row1e5\_entry.config(state="disabled")

print(f"User's answer = {tmpAnswer}")

greenList, yellowList, greyList = self.fnStoreUserInputList(

tmpAnswer.upper())

print(greenList)

print(yellowList)

print(greyList)

# If we write loop like this, i becomes value rather than position.

for i in greenList:

if i == 1:

row1e1\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 2:

row1e2\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 3:

row1e3\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 4:

row1e4\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 5:

row1e5\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

for i in yellowList:

if i == 1:

row1e1\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 2:

row1e2\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 3:

row1e3\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 4:

row1e4\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 5:

row1e5\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

for i in greyList:

if i == 1:

row1e1\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 2:

row1e2\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 3:

row1e3\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 4:

row1e4\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 5:

row1e5\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

# Checking if answer is correct or not.

if len(greenList) == 5:

submitButton.destroy()

tmpLabelcorrect = Label(

wordle\_screen, text="Congratulations! Your Answer is CORRECT.", fg="green", font=("calibri", 11))

tmpLabelcorrect.pack()

# Answer submission button...

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

btnNextLevel = Button(wordle\_screen, text="Next Level", font="buttonFont",

width=10, height=1, bg="dark green", fg="magenta")

btnNextLevel.place(relx=0.0, rely=1.0, anchor='sw')

btnNextLevel.pack()

btnNextLevel.config(command=self.fnNextLevel)

else:

tmpLabel = Label(

wordle\_screen, text="Incorrect Answer. Try Again.", fg="green", font=("calibri", 11))

tmpLabel.pack()

self.tempCount += 1

# --- For second attempt ----

elif self.tempCount == 2:

tmpAnswer = row2e1\_entry.get() + row2e2\_entry.get() + row2e3\_entry.get() + \

row2e4\_entry.get() + row2e5\_entry.get()

row2e1\_entry.config(state="disabled")

row2e2\_entry.config(state="disabled")

row2e3\_entry.config(state="disabled")

row2e4\_entry.config(state="disabled")

row2e5\_entry.config(state="disabled")

print(f"User's answer = {tmpAnswer}")

greenList, yellowList, greyList = self.fnStoreUserInputList(

tmpAnswer.upper())

print(greenList)

print(yellowList)

print(greyList)

for i in greenList:

if i == 1:

row2e1\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 2:

row2e2\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 3:

row2e3\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 4:

row2e4\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 5:

row2e5\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

for i in yellowList:

if i == 1:

row2e1\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 2:

row2e2\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 3:

row2e3\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 4:

row2e4\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 5:

row2e5\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

for i in greyList:

if i == 1:

row2e1\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 2:

row2e2\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 3:

row2e3\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 4:

row2e4\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 5:

row2e5\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

# Checking if answer is correct or not.

tmpLabel.destroy()

if len(greenList) == 5:

submitButton.destroy()

tmpLabelcorrect = Label(

wordle\_screen, text="Congratulations! Your Answer is CORRECT.", fg="green", font=("calibri", 11))

tmpLabelcorrect.pack()

# Answer submission button...

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

btnNextLevel = Button(wordle\_screen, text="Next Level", font="buttonFont",

width=10, height=1, bg="dark green", fg="magenta")

btnNextLevel.place(relx=0.0, rely=1.0, anchor='sw')

btnNextLevel.pack()

btnNextLevel.config(command=self.fnNextLevel)

else:

tmpLabel.destroy()

tmpLabel = Label(

wordle\_screen, text="Incorrect Answer. Try Again.", fg="green", font=("calibri", 11))

tmpLabel.pack()

self.tempCount += 1

# ----- For third attempt ------

elif self.tempCount == 3:

tmpAnswer = row3e1\_entry.get() + row3e2\_entry.get() + row3e3\_entry.get() + \

row3e4\_entry.get() + row3e5\_entry.get()

row3e1\_entry.config(state="disabled")

row3e2\_entry.config(state="disabled")

row3e3\_entry.config(state="disabled")

row3e4\_entry.config(state="disabled")

row3e5\_entry.config(state="disabled")

print(f"User's answer = {tmpAnswer}")

greenList, yellowList, greyList = self.fnStoreUserInputList(

tmpAnswer.upper())

print(greenList)

print(yellowList)

print(greyList)

for i in greenList:

if i == 1:

row3e1\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 2:

row3e2\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 3:

row3e3\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 4:

row3e4\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 5:

row3e5\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

for i in yellowList:

if i == 1:

row3e1\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 2:

row3e2\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 3:

row3e3\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 4:

row3e4\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 5:

row3e5\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

for i in greyList:

if i == 1:

row3e1\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 2:

row3e2\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 3:

row3e3\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 4:

row3e4\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 5:

row3e5\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

# Checking if answer is correct or not.

tmpLabel.destroy()

if len(greenList) == 5:

submitButton.destroy()

tmpLabelcorrect = Label(

wordle\_screen, text="Congratulations! Your Answer is CORRECT.", fg="green", font=("calibri", 11))

tmpLabelcorrect.pack()

# Answer submission button...

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

btnNextLevel = Button(wordle\_screen, text="Next Level", font="buttonFont",

width=10, height=1, bg="dark green", fg="magenta")

btnNextLevel.place(relx=0.0, rely=1.0, anchor='sw')

btnNextLevel.pack()

btnNextLevel.config(command=self.fnNextLevel)

else:

tmpLabel.destroy()

tmpLabel = Label(

wordle\_screen, text="Incorrect Answer. Try Again.", fg="green", font=("calibri", 11))

tmpLabel.pack()

self.tempCount += 1

# Checking for FORTH attempt ----.

elif self.tempCount == 4:

tmpAnswer = row4e1\_entry.get() + row4e2\_entry.get() + row4e3\_entry.get() + \

row4e4\_entry.get() + row4e5\_entry.get()

row4e1\_entry.config(state="disabled")

row4e2\_entry.config(state="disabled")

row4e3\_entry.config(state="disabled")

row4e4\_entry.config(state="disabled")

row4e5\_entry.config(state="disabled")

print(f"User's answer = {tmpAnswer}")

greenList, yellowList, greyList = self.fnStoreUserInputList(

tmpAnswer.upper())

print(greenList)

print(yellowList)

print(greyList)

for i in greenList:

if i == 1:

row4e1\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 2:

row4e2\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 3:

row4e3\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 4:

row4e4\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 5:

row4e5\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

for i in yellowList:

if i == 1:

row4e1\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 2:

row4e2\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 3:

row4e3\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 4:

row4e4\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 5:

row4e5\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

for i in greyList:

if i == 1:

row4e1\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 2:

row4e2\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 3:

row4e3\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 4:

row4e4\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 5:

row4e5\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

# Checking if answer is correct or not.

tmpLabel.destroy()

if len(greenList) == 5:

submitButton.destroy()

tmpLabelcorrect = Label(

wordle\_screen, text="Congratulations! Your Answer is CORRECT.", fg="green", font=("calibri", 11))

tmpLabelcorrect.pack()

# Answer submission button...

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

btnNextLevel = Button(wordle\_screen, text="Next Level", font="buttonFont",

width=10, height=1, bg="dark green", fg="magenta")

btnNextLevel.place(relx=0.0, rely=1.0, anchor='sw')

btnNextLevel.pack()

btnNextLevel.config(command=self.fnNextLevel)

else:

tmpLabel.destroy()

tmpLabel = Label(

wordle\_screen, text="Incorrect Answer. Try Again.", fg="green", font=("calibri", 11))

tmpLabel.pack()

self.tempCount += 1

# Checking for FIFTH attempt ----.

elif self.tempCount == 5:

tmpAnswer = row5e1\_entry.get() + row5e2\_entry.get() + row5e3\_entry.get() + \

row5e4\_entry.get() + row5e5\_entry.get()

row5e1\_entry.config(state="disabled")

row5e2\_entry.config(state="disabled")

row5e3\_entry.config(state="disabled")

row5e4\_entry.config(state="disabled")

row5e5\_entry.config(state="disabled")

print(f"User's answer = {tmpAnswer}")

greenList, yellowList, greyList = self.fnStoreUserInputList(

tmpAnswer.upper())

print(greenList)

print(yellowList)

print(greyList)

for i in greenList:

if i == 1:

row5e1\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 2:

row5e2\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 3:

row5e3\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 4:

row5e4\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 5:

row5e5\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

for i in yellowList:

if i == 1:

row5e1\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 2:

row5e2\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 3:

row5e3\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 4:

row5e4\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 5:

row5e5\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

for i in greyList:

if i == 1:

row5e1\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 2:

row5e2\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 3:

row5e3\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 4:

row5e4\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 5:

row5e5\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

# Checking if answer is correct or not.

tmpLabel.destroy()

if len(greenList) == 5:

submitButton.destroy()

tmpLabelcorrect = Label(

wordle\_screen, text="Congratulations! Your Answer is CORRECT.", fg="green", font=("calibri", 11))

tmpLabelcorrect.pack()

# Answer submission button...

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

btnNextLevel = Button(wordle\_screen, text="Next Level", font="buttonFont",

width=10, height=1, bg="dark green", fg="magenta")

btnNextLevel.place(relx=0.0, rely=1.0, anchor='sw')

btnNextLevel.pack()

btnNextLevel.config(command=self.fnNextLevel)

else:

tmpLabel.destroy()

tmpLabel = Label(

wordle\_screen, text="Incorrect Answer. Try Again.", fg="green", font=("calibri", 11))

tmpLabel.pack()

self.tempCount += 1

# Checking for SIXTH attempt ----.

elif self.tempCount == 6:

tmpAnswer = row6e1\_entry.get() + row6e2\_entry.get() + row6e3\_entry.get() + \

row6e4\_entry.get() + row6e5\_entry.get()

row6e1\_entry.config(state="disabled")

row6e2\_entry.config(state="disabled")

row6e3\_entry.config(state="disabled")

row6e4\_entry.config(state="disabled")

row6e5\_entry.config(state="disabled")

print(f"User's answer = {tmpAnswer}")

greenList, yellowList, greyList = self.fnStoreUserInputList(

tmpAnswer.upper())

print(greenList)

print(yellowList)

print(greyList)

for i in greenList:

if i == 1:

row6e1\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 2:

row6e2\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 3:

row6e3\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 4:

row6e4\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

if i == 5:

row6e5\_entry.configure(

disabledbackground="green", state="disabled", disabledforeground="black")

for i in yellowList:

if i == 1:

row6e1\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 2:

row6e2\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 3:

row6e3\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 4:

row6e4\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

if i == 5:

row6e5\_entry.configure(

disabledbackground="yellow", state="disabled", disabledforeground="black")

for i in greyList:

if i == 1:

row6e1\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 2:

row6e2\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 3:

row6e3\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 4:

row6e4\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

if i == 5:

row6e5\_entry.configure(

disabledbackground="grey", state="disabled", disabledforeground="black")

# Checking if answer is correct or not.

tmpLabel.destroy()

if len(greenList) == 5:

submitButton.destroy()

tmpLabelcorrect = Label(

wordle\_screen, text="Congratulations! Your Answer is CORRECT.", fg="green", font=("calibri", 11))

tmpLabelcorrect.pack()

# Answer submission button...

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

btnNextLevel = Button(wordle\_screen, text="Next Level", font="buttonFont",

width=10, height=1, bg="dark green", fg="magenta")

btnNextLevel.place(relx=0.0, rely=1.0, anchor='sw')

btnNextLevel.pack()

btnNextLevel.config(command=self.fnNextLevel)

else:

# Destroying previous label(because I can't print label six times. So destroying previous label and then overriding next label.)

submitButton.destroy()

tmpLabel.destroy()

tmpLabel = Label(

wordle\_screen, text="Incorrect Answer. You lose this game. No Attempts Remain.", fg="green", font=("calibri", 11))

tmpLabel.pack()

# Calling function to get correct answer.

self.finalWord = self.getCorrectAnswer()

# Printing correct answer.

Label(wordle\_screen, text="Correct Answer Is...",

fg="blue", font=("calibri", 15)).pack()

Label(wordle\_screen, text=self.finalWord,

fg="blue", font=("calibri", 15)).pack()

# Answer submission button...

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

btnNextLevel = Button(wordle\_screen, text="Next Level", font="buttonFont",

width=10, height=1, bg="dark green", fg="magenta")

btnNextLevel.place(relx=0.0, rely=1.0, anchor='sw')

btnNextLevel.pack()

btnNextLevel.config(command=self.fnNextLevel)

self.tempCount = 1

def fnNextLevel(self):

global wordle\_screen

wordle\_screen.destroy()

self.wordle\_test()

class GameMainClass(LoginClass, UserAttemptClass):

# Designing Main(first) window

def main\_account\_screen(self):

global main\_screen

main\_screen = Tk()

main\_screen.geometry("350x500")

main\_screen.title("Account Login")

# Added for Wordle....

iframe = Frame(main\_screen)

iframe.pack()

self.topframe = Frame(main\_screen)

self.topframe.pack()

mframe = Frame(main\_screen)

mframe.pack()

L1 = Label(iframe, font=("Apple, 10"))

L1.pack()

# Calling method of RegisterClass to create UI of "WORDLE" word so no need to repeat this task again and again.

self.fnUIWordle()

L1 = Label(mframe, font=("Apple, 10"))

L1.pack()

# End of wrodle UI....

Label(text="Select Your Choice", bg="blue", fg="white",

width="300", height="2", font=("Calibri", 13)).pack()

Label(text="").pack()

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

Button(text="Login", font="buttonFont", height="2",

width="20", bg="blue", fg="magenta", command=lambda: self.login()).pack()

Label(text="").pack()

button = Button(text="Register", font="buttonFont", height="2",

width="20", bg="blue", fg="magenta", command=lambda: self.register()).pack()

main\_screen.mainloop()

# Starting wordle GAME UI--------

# ----- Main wordle game UI for every attemts ------.

def wordle\_test(self):

self.tempCount = 1

from tkinter import LEFT, RIGHT, Button, Entry, Frame, Label, StringVar, Tk, Toplevel, font

global submitButton

global wordle\_screen

wordle\_screen = Tk()

wordle\_screen.title("Wordle Game Main Screen")

wordle\_screen.geometry("400x650")

# ---Frames are added ---.

mframe = Frame(wordle\_screen)

mframe.pack()

self.topframe = Frame(wordle\_screen)

self.topframe.pack()

iframe = Frame(wordle\_screen)

iframe.pack()

hintframe = Frame(wordle\_screen)

hintframe.pack()

frame1 = Frame(wordle\_screen)

frame1.pack()

frame2 = Frame(wordle\_screen)

frame2.pack()

frame3 = Frame(wordle\_screen)

frame3.pack()

frame4 = Frame(wordle\_screen)

frame4.pack()

frame5 = Frame(wordle\_screen)

frame5.pack()

frame6 = Frame(wordle\_screen)

frame6.pack()

bottomframe = Frame(wordle\_screen)

bottomframe.pack()

nextlevelframe = Frame(wordle\_screen)

nextlevelframe.pack()

L1 = Label(mframe, font=("Apple, 20"), text="")

L1.pack()

# Calling method of RegisterClass to create UI of "WORDLE" word so no need to repeat this task again and again.

self.fnUIWordle()

# Printing username in top right corner.

tmpUserPrint = "Welcome " + '"' + self.username + '"'

L7 = Label(wordle\_screen, text=tmpUserPrint,

fg="#101057", font=("Calibri", 16))

L7.place(relx=1.0, rely=0.0, anchor='ne')

Label(iframe, text="Please enter details below", bg="blue",

fg="white", width="300", height="1", font=("Calibri", 13)).pack()

Label(iframe, text="").pack(pady=4)

# Logout button to configure.

def fnLogout():

global wordle\_screen

wordle\_screen.destroy()

logoutButton = Button(wordle\_screen, font="buttonFont", text="Logout",

fg="magenta", bg="#f7051d", borderwidth=2, relief="solid", command=fnLogout)

logoutButton.place(relx=0.05, rely=0.97, anchor='sw')

# End of Logout button code.

# Calling this function to get New Final Word for this level.

self.finalWord = self.getNewFinalWord()

hint = self.finalWord[0]

finalhint = "Today's word start from " + '"' + hint + '"'

self.hintLabel = Label(wordle\_screen, text="Hint", width=6,

fg="magenta", bg="white", height="1", font=("Calibri", 13))

self.hintLabel.place(rely=0.97, relx=0.97, x=0, y=0, anchor=SE)

self.hintDisplayLabel = Label(wordle\_screen)

self.hintDisplayLabel.place(rely=0.97, relx=0.80, x=0, y=0, anchor=SE)

def on\_enter(event):

self.hintDisplayLabel.configure(

text=finalhint, fg="magenta", bg="white", height="1", font=("Calibri", 13))

def on\_leave(enter):

self.hintDisplayLabel.configure(text="", bg="#fafaf2")

# We can bind the key event using the Binding method in a tkinter application.

self.hintLabel.bind("<Enter>", on\_enter)

self.hintLabel.bind("<Leave>", on\_leave)

# Whenever the key will be triggered, it will call a handler that will raise the specific operation for the key event.

# End of the hint code.

# --- Starting Wordle Game UI for FIRST row. ---.

global row1e1\_entry

global row1e2\_entry

global row1e3\_entry

global row1e4\_entry

global row1e5\_entry

self.row1e1 = StringVar()

self.row1e2 = StringVar()

self.row1e3 = StringVar()

self.row1e4 = StringVar()

self.row1e5 = StringVar()

def caps(event):

self.row1e1.set(self.row1e1.get().upper())

self.row1e2.set(self.row1e2.get().upper())

self.row1e3.set(self.row1e3.get().upper())

self.row1e4.set(self.row1e4.get().upper())

self.row1e5.set(self.row1e5.get().upper())

self.row2e1.set(self.row2e1.get().upper())

self.row2e2.set(self.row2e2.get().upper())

self.row2e3.set(self.row2e3.get().upper())

self.row2e4.set(self.row2e4.get().upper())

self.row2e5.set(self.row2e5.get().upper())

self.row3e1.set(self.row3e1.get().upper())

self.row3e2.set(self.row3e2.get().upper())

self.row3e3.set(self.row3e3.get().upper())

self.row3e4.set(self.row3e4.get().upper())

self.row3e5.set(self.row3e5.get().upper())

self.row4e1.set(self.row4e1.get().upper())

self.row4e2.set(self.row4e2.get().upper())

self.row4e3.set(self.row4e3.get().upper())

self.row4e4.set(self.row4e4.get().upper())

self.row4e5.set(self.row4e5.get().upper())

self.row5e1.set(self.row5e1.get().upper())

self.row5e2.set(self.row5e2.get().upper())

self.row5e3.set(self.row5e3.get().upper())

self.row5e4.set(self.row5e4.get().upper())

self.row5e5.set(self.row5e5.get().upper())

self.row6e1.set(self.row6e1.get().upper())

self.row6e2.set(self.row6e2.get().upper())

self.row6e3.set(self.row6e3.get().upper())

self.row6e4.set(self.row6e4.get().upper())

self.row6e5.set(self.row6e5.get().upper())

row1e1\_entry = Entry(frame1, textvariable=self.row1e1, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row1e1\_entry.pack(side=LEFT, padx=5, pady=5)

row1e2\_entry = Entry(frame1, textvariable=self.row1e2, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row1e2\_entry.pack(side=LEFT, padx=5, pady=5)

row1e3\_entry = Entry(frame1, textvariable=self.row1e3, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row1e3\_entry.pack(side=LEFT, padx=5, pady=5)

row1e4\_entry = Entry(frame1, textvariable=self.row1e4, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row1e4\_entry.pack(side=LEFT, padx=5, pady=5)

row1e5\_entry = Entry(frame1, textvariable=self.row1e5, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row1e5\_entry.pack(side=LEFT)

row1e1\_entry.bind('<KeyRelease>', caps)

row1e2\_entry.bind('<KeyRelease>', caps)

row1e3\_entry.bind('<KeyRelease>', caps)

row1e4\_entry.bind('<KeyRelease>', caps)

row1e5\_entry.bind('<KeyRelease>', caps)

# --- Starting Wordle Game UI for SECOND row. ---.

global row2e1\_entry

global row2e2\_entry

global row2e3\_entry

global row2e4\_entry

global row2e5\_entry

self.row2e1 = StringVar()

self.row2e2 = StringVar()

self.row2e3 = StringVar()

self.row2e4 = StringVar()

self.row2e5 = StringVar()

row2e1\_entry = Entry(frame2, textvariable=self.row2e1, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row2e1\_entry.pack(side=LEFT, padx=5, pady=5)

row2e2\_entry = Entry(frame2, textvariable=self.row2e2, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row2e2\_entry.pack(side=LEFT, padx=5, pady=5)

row2e3\_entry = Entry(frame2, textvariable=self.row2e3, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row2e3\_entry.pack(side=LEFT, padx=5, pady=5)

row2e4\_entry = Entry(frame2, textvariable=self.row2e4, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row2e4\_entry.pack(side=LEFT, padx=5, pady=5)

row2e5\_entry = Entry(frame2, textvariable=self.row2e5, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row2e5\_entry.pack(side=LEFT)

row2e1\_entry.bind('<KeyRelease>', caps)

row2e2\_entry.bind('<KeyRelease>', caps)

row2e3\_entry.bind('<KeyRelease>', caps)

row2e4\_entry.bind('<KeyRelease>', caps)

row2e5\_entry.bind('<KeyRelease>', caps)

# --- Starting Wordle Game UI for THIRD row. ---.

global row3e1\_entry

global row3e2\_entry

global row3e3\_entry

global row3e4\_entry

global row3e5\_entry

self.row3e1 = StringVar()

self.row3e2 = StringVar()

self.row3e3 = StringVar()

self.row3e4 = StringVar()

self.row3e5 = StringVar()

row3e1\_entry = Entry(frame3, textvariable=self.row3e1, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row3e1\_entry.pack(side=LEFT, padx=5, pady=5)

row3e2\_entry = Entry(frame3, textvariable=self.row3e2, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row3e2\_entry.pack(side=LEFT, padx=5, pady=5)

row3e3\_entry = Entry(frame3, textvariable=self.row3e3, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row3e3\_entry.pack(side=LEFT, padx=5, pady=5)

row3e4\_entry = Entry(frame3, textvariable=self.row3e4, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row3e4\_entry.pack(side=LEFT, padx=5, pady=5)

row3e5\_entry = Entry(frame3, textvariable=self.row3e5, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row3e5\_entry.pack(side=LEFT)

row3e1\_entry.bind('<KeyRelease>', caps)

row3e2\_entry.bind('<KeyRelease>', caps)

row3e3\_entry.bind('<KeyRelease>', caps)

row3e4\_entry.bind('<KeyRelease>', caps)

row3e5\_entry.bind('<KeyRelease>', caps)

# --- Starting Wordle Game UI for FORTH row. ---.

global row4e1\_entry

global row4e2\_entry

global row4e3\_entry

global row4e4\_entry

global row4e5\_entry

self.row4e1 = StringVar()

self.row4e2 = StringVar()

self.row4e3 = StringVar()

self.row4e4 = StringVar()

self.row4e5 = StringVar()

row4e1\_entry = Entry(frame4, textvariable=self.row4e1, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row4e1\_entry.pack(side=LEFT, padx=5, pady=5)

row4e2\_entry = Entry(frame4, textvariable=self.row4e2, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row4e2\_entry.pack(side=LEFT, padx=5, pady=5)

row4e3\_entry = Entry(frame4, textvariable=self.row4e3, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row4e3\_entry.pack(side=LEFT, padx=5, pady=5)

row4e4\_entry = Entry(frame4, textvariable=self.row4e4, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row4e4\_entry.pack(side=LEFT, padx=5, pady=5)

row4e5\_entry = Entry(frame4, textvariable=self.row4e5, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row4e5\_entry.pack(side=LEFT)

row4e1\_entry.bind('<KeyRelease>', caps)

row4e2\_entry.bind('<KeyRelease>', caps)

row4e3\_entry.bind('<KeyRelease>', caps)

row4e4\_entry.bind('<KeyRelease>', caps)

row4e5\_entry.bind('<KeyRelease>', caps)

# --- Starting Wordle Game UI for FIFTH row. ---.

global row5e1\_entry

global row5e2\_entry

global row5e3\_entry

global row5e4\_entry

global row5e5\_entry

self.row5e1 = StringVar()

self.row5e2 = StringVar()

self.row5e3 = StringVar()

self.row5e4 = StringVar()

self.row5e5 = StringVar()

row5e1\_entry = Entry(frame5, textvariable=self.row5e1, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row5e1\_entry.pack(side=LEFT, padx=5, pady=5)

row5e2\_entry = Entry(frame5, textvariable=self.row5e2, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row5e2\_entry.pack(side=LEFT, padx=5, pady=5)

row5e3\_entry = Entry(frame5, textvariable=self.row5e3, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row5e3\_entry.pack(side=LEFT, padx=5, pady=5)

row5e4\_entry = Entry(frame5, textvariable=self.row5e4, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row5e4\_entry.pack(side=LEFT, padx=5, pady=5)

row5e5\_entry = Entry(frame5, textvariable=self.row5e5, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row5e5\_entry.pack(side=LEFT)

row5e1\_entry.bind('<KeyRelease>', caps)

row5e2\_entry.bind('<KeyRelease>', caps)

row5e3\_entry.bind('<KeyRelease>', caps)

row5e4\_entry.bind('<KeyRelease>', caps)

row5e5\_entry.bind('<KeyRelease>', caps)

# --- Starting Wordle Game UI for SIXTH row. ---.

global row6e1\_entry

global row6e2\_entry

global row6e3\_entry

global row6e4\_entry

global row6e5\_entry

self.row6e1 = StringVar()

self.row6e2 = StringVar()

self.row6e3 = StringVar()

self.row6e4 = StringVar()

self.row6e5 = StringVar()

row6e1\_entry = Entry(frame6, textvariable=self.row6e1, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row6e1\_entry.pack(side=LEFT, padx=5, pady=5)

row6e2\_entry = Entry(frame6, textvariable=self.row6e2, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row6e2\_entry.pack(side=LEFT, padx=5, pady=5)

row6e3\_entry = Entry(frame6, textvariable=self.row6e3, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row6e3\_entry.pack(side=LEFT, padx=5, pady=5)

row6e4\_entry = Entry(frame6, textvariable=self.row6e4, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row6e4\_entry.pack(side=LEFT, padx=5, pady=5)

row6e5\_entry = Entry(frame6, textvariable=self.row6e5, width=2, font=(

'Geogia 20'), bg='black', fg='white', justify='center')

row6e5\_entry.pack(side=LEFT)

row6e1\_entry.bind('<KeyRelease>', caps)

row6e2\_entry.bind('<KeyRelease>', caps)

row6e3\_entry.bind('<KeyRelease>', caps)

row6e4\_entry.bind('<KeyRelease>', caps)

row6e5\_entry.bind('<KeyRelease>', caps)

# Answer submission button...

# This line is to add style to button texts.

buttonFont = font.Font(size=12, weight='bold')

submitButton = Button(bottomframe, text="Submit", font="buttonFont",

width=10, height=1, bg="blue", fg="magenta")

submitButton.pack(pady=20)

submitButton.config(command=self.fnCheckAttemptNo)

wordle\_screen.mainloop()

# Calling main account function. Beginning Page.8

tmpObj = GameMainClass()

tmpObj.main\_account\_screen()

**Conclusion**

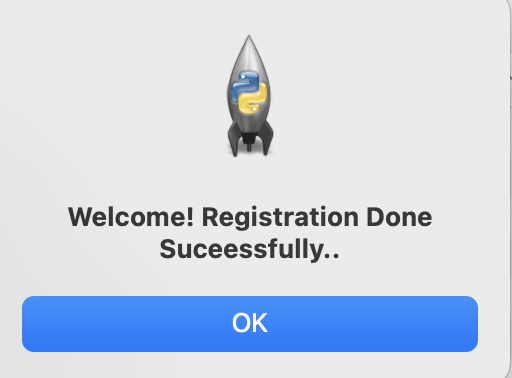
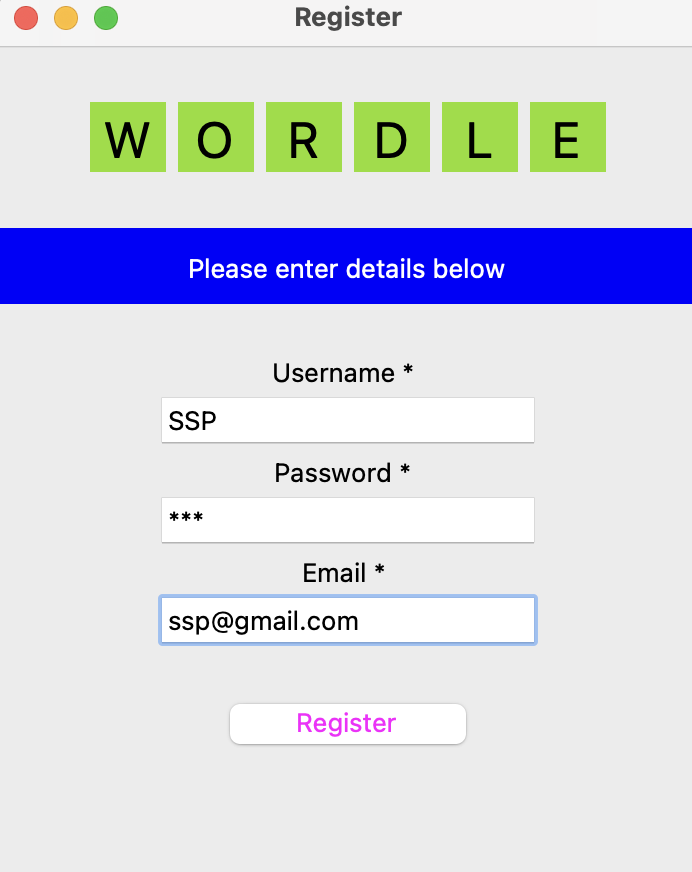
The Wordle game is designed in python programming language and uses supporting packages and libraries. Players had to make guesses in a minimum of attempts in order to keep the streak of winning. The system picks a brand daily word using random selection from the file. The very first attempt the player participates in is crucial. All the best!

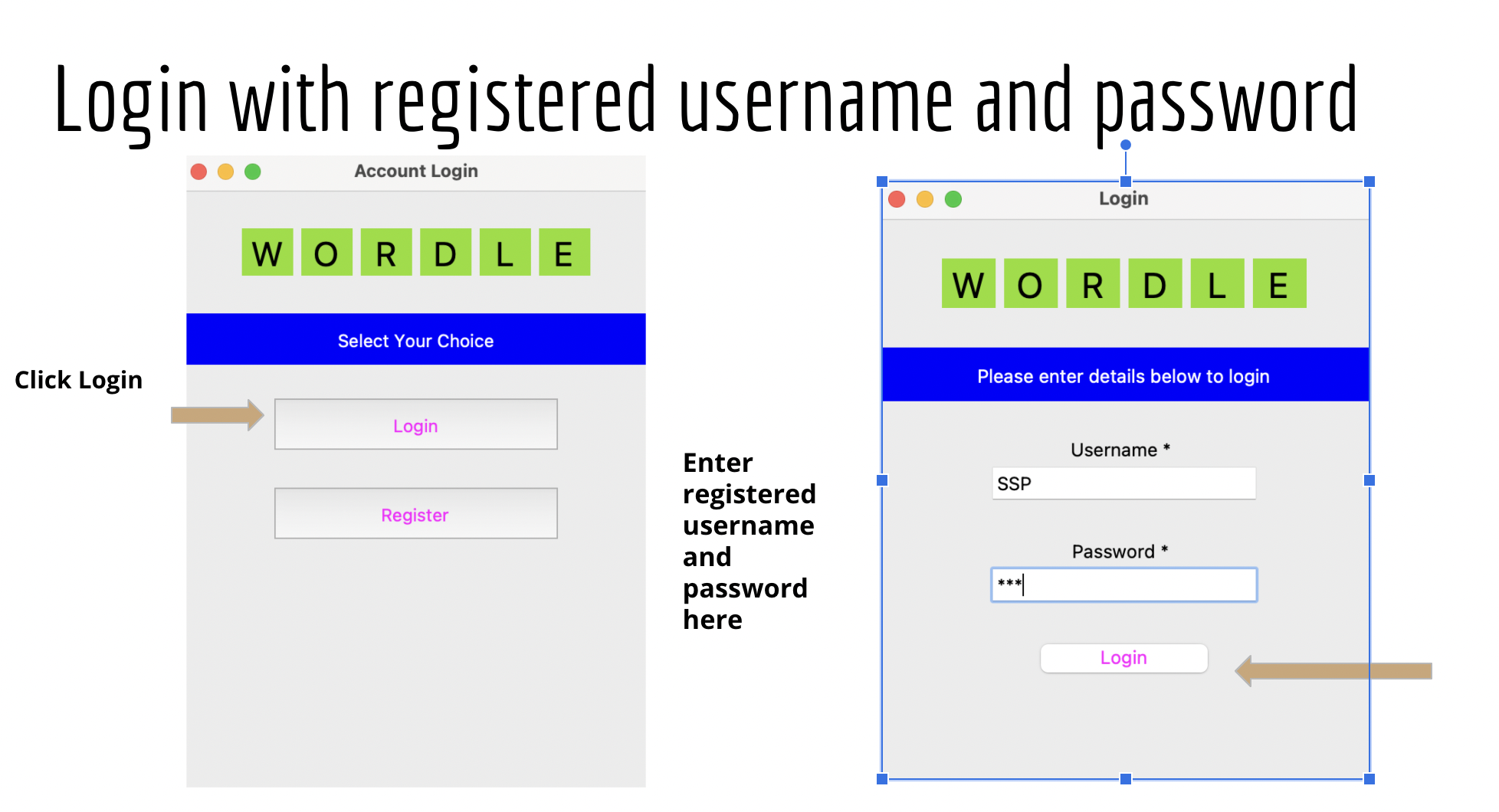
**Screenshots of game(step by step):**

The first window pops up in front of you when you open game file is Login/Register as shown below;



You need to register with your username, password and email id

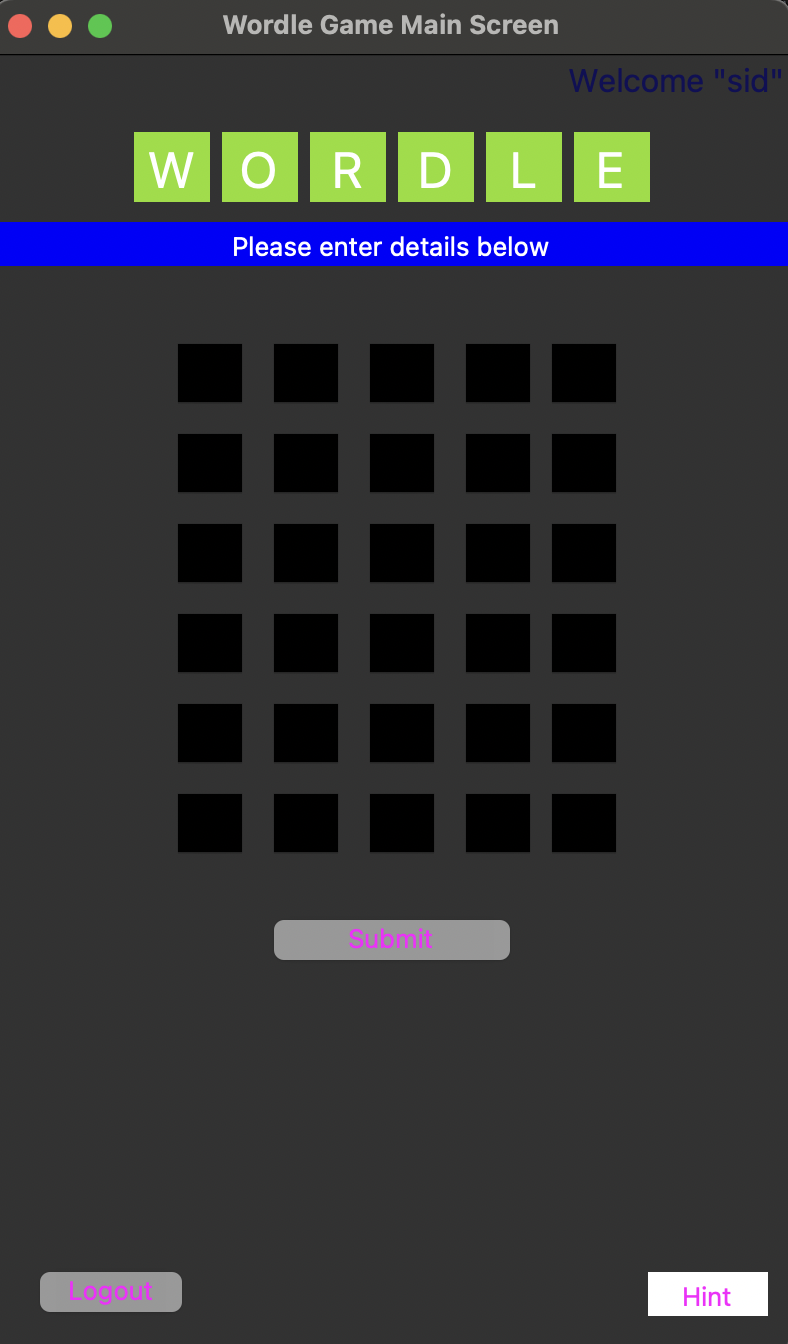




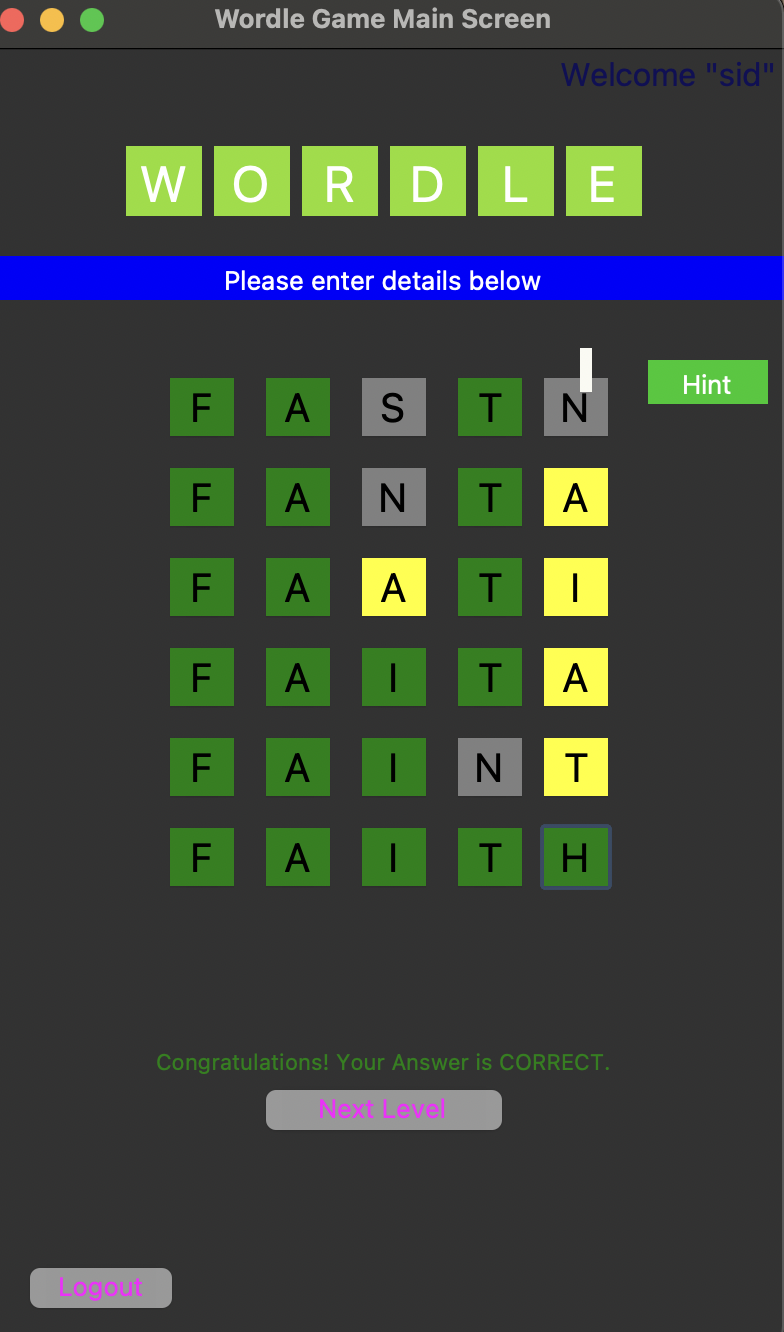


**How to play wordle game explained using screenshots:**

1. Start with any word of the length of box provided in a row



1. Try to guess the secret word

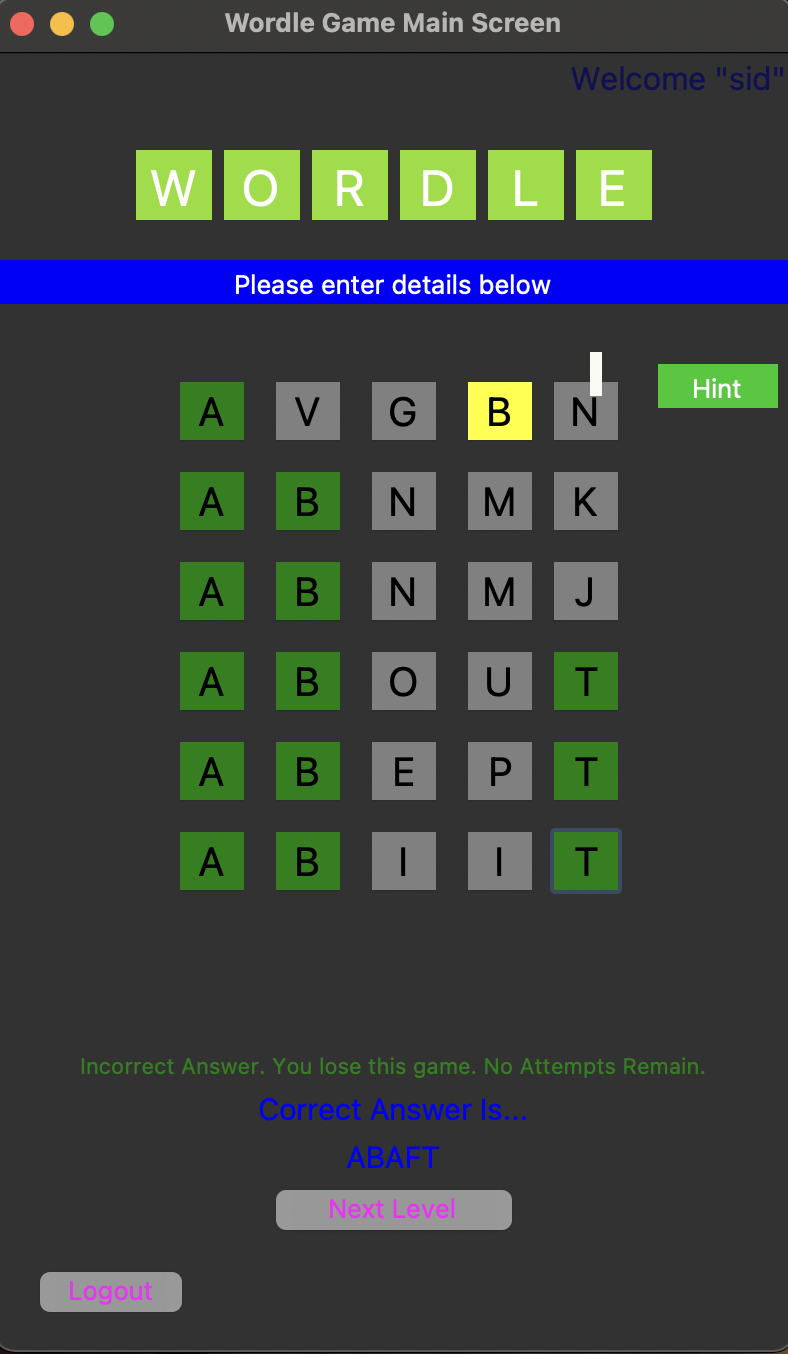


After successfully guessing the word

**“Congratulations! Your Answer is correct.”**message

will pop up.

1. To play again click next level
2. **If you fail to guess the correct word in 6 attempts :** Click Next level to play again or click Logout to exit the game.



**The End.**