**SPEED TYPING TEST**

*A*

*Mini Project Report*

*Submitted in partial fulfilment of the*

*Requirements for the award of the Degree of*

**BACHELOR OF ENGINEERING**

IN

**INFORMATION TECHNOLOGY**

By

**Sreenija Alaka\_ 1602-20-737-044**

**Srija Dandu \_ 1602-20-737-045**

**Shruthi Bhavana\_1602-20-737-047**

****

**Department of Information Technology**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University and Approved by AICTE)**

**Ibrahimbagh, Hyderabad-31**

**2022**

**Vasavi College of Engineering (Autonomous)**

**(Affiliated to Osmania University)**

**Hyderabad-500 031**

**Department of Information Technology**

****

**DECLARATION BY THE CANDIDATE**

We, **A.Sreenja, D.Srija** and **Shruthi Bhavan,** bearing hall ticket numbers, **1602-20-737-044, 1602-20-737-045** and **1602-20-737-047**, hereby declare that the project report entitled **“SPEED TYPING TEST”** is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by us and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

**A.Sreenija**

**1602-20-737-044**

**D.Srija**

**1602-20-737-045**

**Shruthi Bhavana**

**1602-20-737-047**

(Faculty In-Charge) (Head,Dept of IT)

**Acknowledgement**

We extend our sincere thanks to Dr. S. V. Ramana, Principal, Vasavi College of Engineering for his encouragement.

We express our sincere gratitude to Dr. K. Ram Mohan Rao, Professor & Head, Department of Information Technology, Vasavi College of Engineering, for introducing the Mini-Project module in our curriculum, and also for his suggestions, motivation, and co-operation for the successful completion of our Mini Project.

We also want to thank and convey our gratitude towards our mini project coordinators DIVYA L and S. RAJYA LAXMI for guiding us in understanding the process of project development & giving us timely suggestions at every phase.

We would also like to sincerely thank the project reviewers for their valuable inputs and suggestions

**Abstract**

* Typing Speed Test application is useful to test/measure the typing speed of a user. Learn typing and find how fast you can type. Letters are highlighted to help you focus on typing.
* You can become a typing master with the help of this app or can play typing games for fun. You need to add the language-specific keyboard to type in that language.
* In this project, you will create a Typing Test, which will allow users to measure their typing speed.
* Implementation of the user level is intended to secure the data from unauthorized third parties so that only users can see the results of learning.
* Furthermore, the application also gives users the opportunity to participate in learning

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Topic** | **Page No.** |
| 1. | Introduction | 6 |
| 2. | Technology | 7 |
| 3. | Design | 8 |
| 4. | Implementation | 14 |
| 5. | Testing | 32 |
| 6. | Additional Knowledge Acquired | 39 |
| 7. | Conclusion and Future work | 40 |
| 8. | References | 41 |

**INTRODUCTION**

Have you ever wished if there is an application to improve your Typing Speed? Here our project helps user to test your typing skills. The User need to Register and login to start the Typing test then the user is redirected to instructions page. When the user clicks the start button he is redirected to the game where he have to play level 1. If he types the given sentence with 100% accuracy and with in 25 seconds he gets to play level 2 and so on.. If not the result is displayed as Accuracy, Time and Words per minute and with a Retry button.

Graphical User Interface :

A graphical user interface (GUI, for short) is used for graphical purpose.. In order to render the graphics, We have to import Tkinter. Python's main graphics library, In most cases, it is installed automatically when you install Python because it is part of the standard library. Once you've done that, you can run the GUI from your terminal

**TECHNOLOGY**

**Hardware Requirements**

* 512 MB RAM
* 2GB HDD
* CORE i5

**Software Requirements**

* Windows XP/ Windows 2000
* PYTHON INTERPRETER

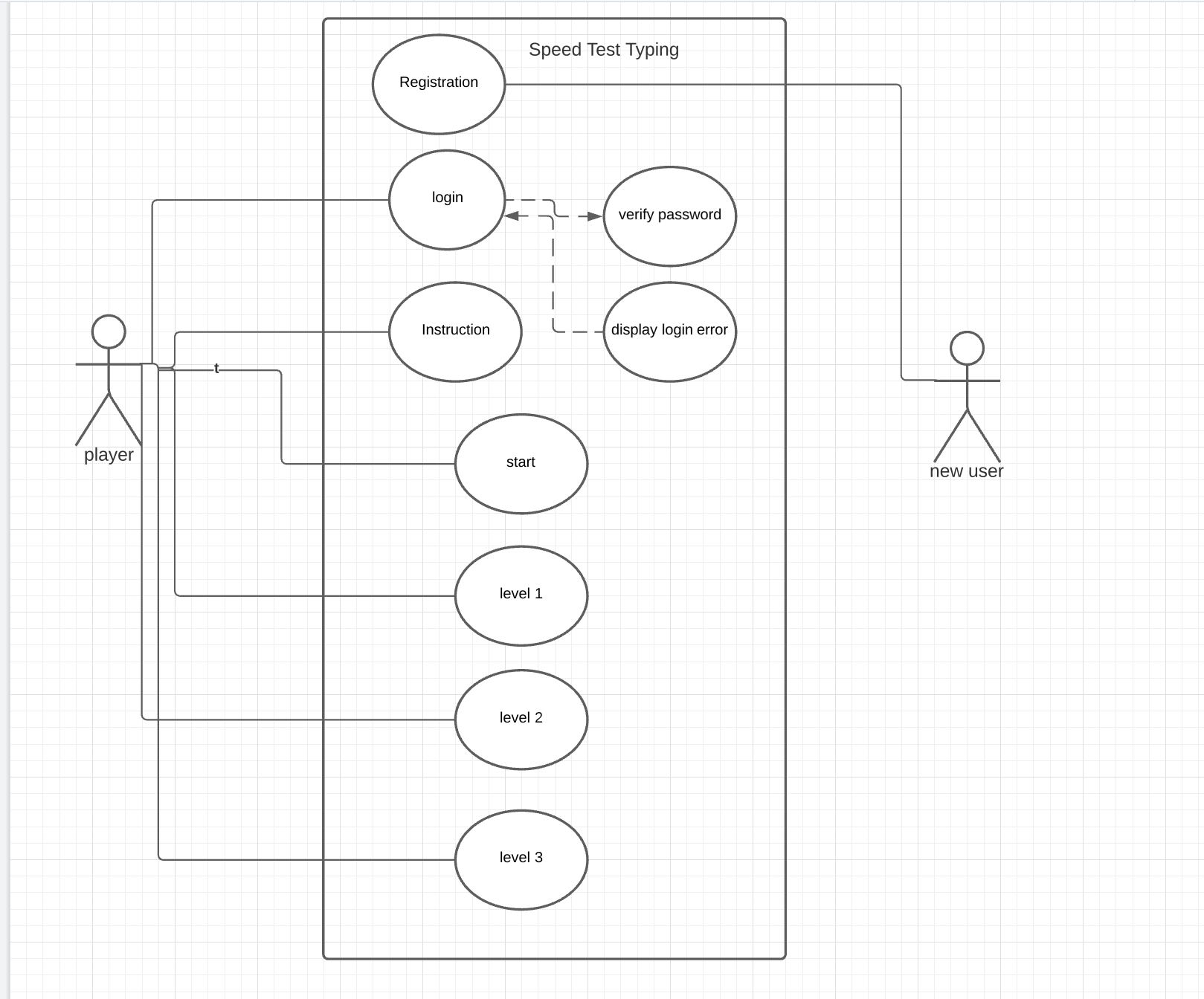
**Packages used**

* Tkinter
* pygame
* random

**PROPOSED WORK**

**a.DESIGN**

**USE CASE DIAGRAM**

****

**Use case descriptions**

**Use Case ID :** UC01

**Name :** Registration

**Actors :** New user

**Description :** Allows new user to register for an account

**Pre-Conditions :** None

**Post-Conditions :** An account is created for the user

**Main Flow :**

|  |  |
| --- | --- |
| **User** | **System** |
| 1.Choose the register option |  |
|  | 2. Prompts for data required for registration |
| 3. Enters the data prompted by the system |  |
|  | 4. Validates user information  If information is valid , creates a new account. |

**Use Case ID:** UC02

**Name:** Login

**Actors:** Current User/ New User

**Description:** Allows registered users to Login

**Pre-conditions:** user should be registered with the system

**Post-conditions:** Starts the game

**Main Flow :**

|  |  |
| --- | --- |
| **User** | **System** |
| 1. Enters username and chooses Login option. |  |
|  | 2.Validates username and password.  2.1) If the user is invalid, display error message and ask the user to register. |

**Use Case ID:** UC03

**Name:** Test

**Actors:** New user/current user

**Description:** Starts typing test

**Pre-conditions:** Starts the game

**Post-conditions:** User types the text given

**Main Flow :**

|  |  |
| --- | --- |
| **User** | **System** |
|  | 1.Shows the text which should be typed |
| 2.User types the text shown by the system |  |

**Use Case ID:** UC04

**Name:** Levels

**Actors:** Current user/New user

**Description:** Increases the level if given condition satisfies.

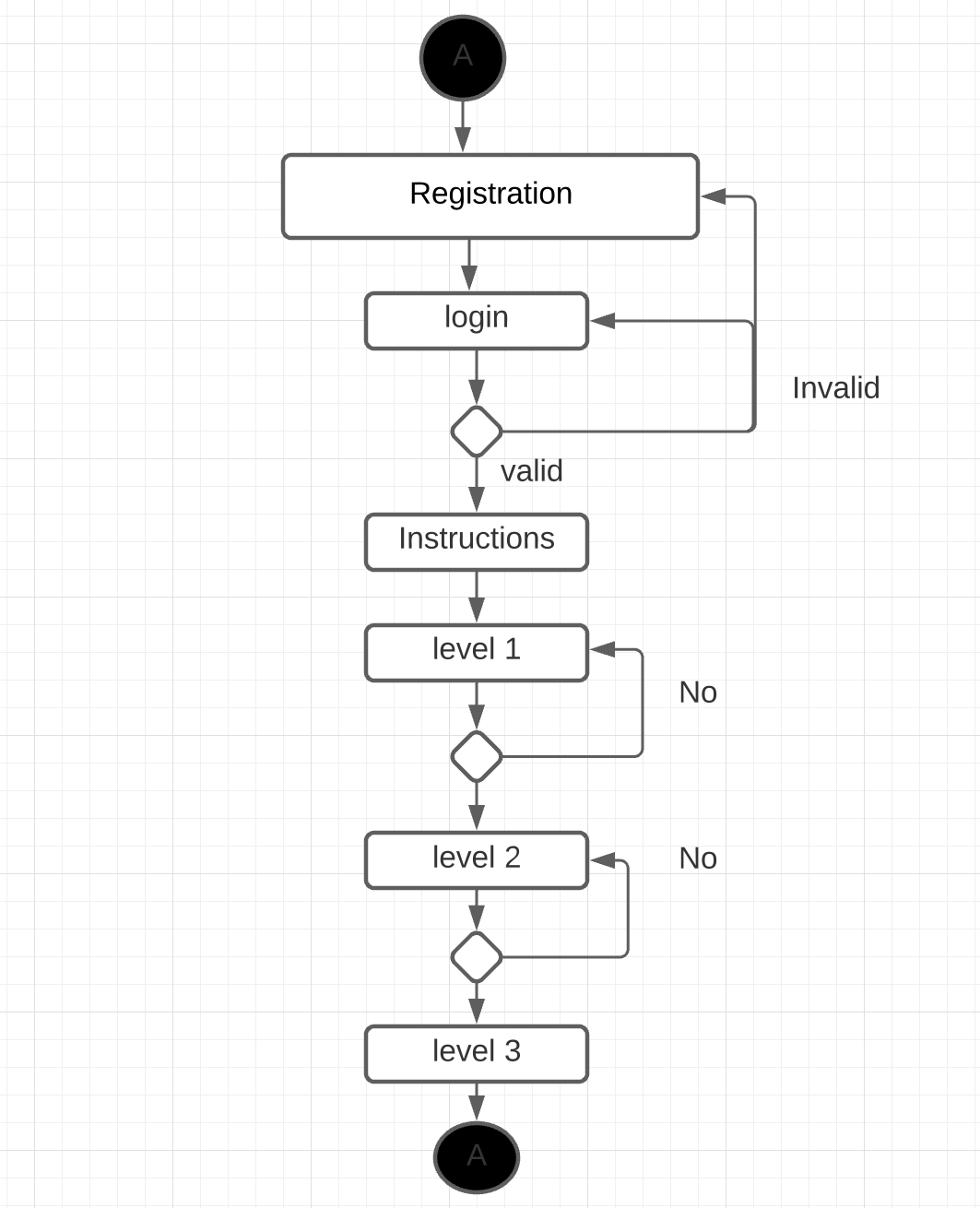
**Pre-conditions:** Starts typing the text

**Post-conditions:** Displays the test result/ increases the level.

**Main Flow :**

|  |  |
| --- | --- |
| **User** | **System** |
|  | 1.Shows wpm, accuracy, time,grade. |
| 2. Starts typing in the next level. |  |

**Activity Diagram:**

****

**Implementation**

#registration,login,game 1

from tkinter import \*

import os

import pygame

from pygame.locals import \*

import sys

import time

import random

def register():

global register\_screen

register\_screen = Toplevel(main\_screen)

register\_screen.title("Register")

register\_screen.geometry("300x250")global username

global password

global username\_entry

global password\_entry

username = StringVar()

password = StringVar()

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

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

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

username\_lable.pack()

username\_entry = Entry(register\_screen, textvariable=username)

username\_entry.pack()

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

password\_lable.pack()

password\_entry = Entry(register\_screen, textvariable=password, show='\*')

password\_entry.pack()

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

Button(register\_screen, text="Register", width=10, height=1, bg="blue", command=register\_user).pack()

def login():

global login\_screen

login\_screen = Toplevel(main\_screen)

login\_screen.title("Login")

login\_screen.geometry("300x250")

Label(login\_screen, text="Please enter details below to login").pack()

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

global username\_verify

global password\_verify

username\_verify = StringVar()

password\_verify = StringVar()

global username\_login\_entry

global password\_login\_entry

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

username\_login\_entry = Entry(login\_screen, textvariable=username\_verify)

username\_login\_entry.pack()

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

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

password\_login\_entry = Entry(login\_screen, textvariable=password\_verify, show='\*')

password\_login\_entry.pack()

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

Button(login\_screen, text="Login", width=10, height=1, command=login\_verify).pack()

def register\_user():

username\_info = username.get()

password\_info = password.get()

file = open(username\_info, "w")

file.write(username\_info + "\n")

file.write(password\_info)

file.close()

username\_entry.delete(0, END)

password\_entry.delete(0, END)

Label(register\_screen, text="Registration Success", fg="green", font=("calibri", 11)).pack()

def login\_verify():

username1 = username\_verify.get()

password1 = password\_verify.get()

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

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

list\_of\_files = os.listdir()

if username1 in list\_of\_files:

file1 = open(username1, "r")

verify = file1.read().splitlines()

if password1 in verify:

# login\_sucess()

instructions()

else:

password\_not\_recognised()

else:

user\_not\_found()

def instructions():

global instruction\_screen

instruction\_screen = Toplevel(main\_screen)

instruction\_screen.geometry("450x300")

instruction\_screen.title("Instructions")

Label(instruction\_screen,text="INSTRUCTIONS\n \* There are 3 levels in this Application.\n \* The Start Button redirects you to Level 1\n \* If you type the Given Sentence with 100% \nAccuracy within 25 seconds you \nget to play level 2 and then level 3\n\* If the above conditions are not satisfied \nyou can retry in the same level until \nthe given conditions are satisfied\n\*ALL THE BEST\*\n", width="300", height="12", font=("Calibri", 12)).pack()

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

Button(instruction\_screen,text="Start Game", bg="blue", height="2", width="20", command=login\_sucess).pack()

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

def login\_sucess():

login\_screen.destroy();

main\_screen.destroy();

class Game:

def \_init\_(self):

self.w = 850

self.h = 500

self.reset = True

self.active = False

self.input\_text = ''

self.word = ''

self.time\_start = 0

self.total\_time = 0

self.accuracy = '0%'

self.results = 'Time:0 Accuracy:0 % Wpm:0 '

self.wpm = 0

self.end = False

self.HEAD\_C = (0, 0, 0) # for colours

self.TEXT\_C = (0, 0, 0)

self.RESULT\_C = (0, 0, 0)

pygame.init()

self.open\_img = pygame.image.load('type-speed-open.png')

self.open\_img = pygame.transform.scale(self.open\_img, (self.w, self.h))

self.bg = pygame.image.load('background.jpg')

self.bg = pygame.transform.scale(self.bg, (900, 600))

self.screen = pygame.display.set\_mode((self.w, self.h))

pygame.display.set\_caption('Type Speed test')

def draw\_text(self, screen, msg, y, fsize, color):

font = pygame.font.Font(None, fsize)

text = font.render(msg, 1, color)

text\_rect = text.get\_rect(center=(self.w / 2, y))

screen.blit(text, text\_rect)

pygame.display.update()

def get\_sentence(self):

f = open('level1.txt').read()

sentences = f.split('\n')

sentence = random.choice(sentences)

return sentence

def show\_results(self, screen):

if (not self.end):

self.total\_time = time.time() - self.time\_start

count = 0

for i, c in enumerate(self.word):

try:

if self.input\_text[i] == c:

count += 1

except:

pass

self.accuracy = count / len(self.word) \* 100

self.wpm = len(self.input\_text) \* 60 / (5 \* self.total\_time)

self.end = True

print(self.total\_time)

self.results = 'Time:' + str(round(self.total\_time)) + " secs Accuracy:" + str(

round(self.accuracy)) + "%" + ' Wpm: ' + str(round(self.wpm))

if (self.accuracy == 100 and self.total\_time < 25) :

level2();

self.time\_img = pygame.image.load('icon.png')

self.time\_img = pygame.transform.scale(self.time\_img, (150, 150))

screen.blit(self.time\_img, (self.w / 2 - 75, self.h - 140))

self.draw\_text(screen, "", self.h - 70, 26, (100, 100, 100))

print(self.results)

pygame.display.update()

def run(self):

self.reset\_game()

self.running = True

while (self.running):

clock = pygame.time.Clock()

self.screen.fill((0, 0, 0), (50, 250, 650, 50))

pygame.draw.rect(self.screen, self.HEAD\_C, (50, 250, 650, 50), 2)

self.draw\_text(self.screen, self.input\_text, 274, 26, (250, 250, 250))

pygame.display.update()

for event in pygame.event.get():

if event.type == QUIT:

self.running = False

sys.exit()

elif event.type == pygame.MOUSEBUTTONUP:

x, y = pygame.mouse.get\_pos()

if (x >= 50 and x <= 650 and y >= 250 and y <= 300):

self.active = True

self.input\_text = ''

self.time\_start = time.time()

if (x >= 310 and x <= 510 and y >= 390 and self.end):

self.reset\_game()

x, y = pygame.mouse.get\_pos()

elif event.type == pygame.KEYDOWN:

if self.active and not self.end:

if event.key == pygame.K\_RETURN:

print(self.input\_text)

self.show\_results(self.screen)

print(self.results)

self.draw\_text(self.screen, self.results, 350, 28, self.RESULT\_C)

self.end = True

elif event.key == pygame.K\_BACKSPACE:

self.input\_text = self.input\_text[:-1]

else:

try:

self.input\_text += event.unicode

except:

pass

pygame.display.update()

clock.tick(60)

def reset\_game(self):

self.screen.blit(self.open\_img, (0, 0))

pygame.display.update()

time.sleep(1)

self.reset = False

self.end = False

self.input\_text = ''

self.word = ''

self.time\_start = 0

self.total\_time = 0

self.wpm = 0

self.word = self.get\_sentence()

if (not self.word): self.reset\_game()

# drawing heading

self.screen.fill((0, 0, 0))

self.screen.blit(self.bg, (0, 0))

msg = "Let's test your typing speed !!"

self.draw\_text(self.screen, msg, 60, 50, self.HEAD\_C)

pygame.draw.rect(self.screen, (255, 192, 25), (50, 250, 650, 50), 2)

self.draw\_text(self.screen, self.word, 200, 28, self.TEXT\_C)

pygame.display.update()

level = 1

def level1():

global level

level = 1

mainLevel()

def level2():

global level

level = 2

mainLevel()

def level3():

global level

level = 3

mainLevel()

def mainLevel():

class Game:

def \_init\_(self):

self.w = 850

self.h = 500

self.reset = True

self.active = False

self.input\_text = ''

self.word = ''

self.time\_start = 0

self.total\_time = 0

self.accuracy = '0%'

self.results = 'Time:0 Accuracy:0 % Wpm:0 '

self.wpm = 0

self.end = False

self.HEAD\_C = (0, 0, 0) # for colours

self.TEXT\_C = (0, 0, 0)

self.RESULT\_C = (0, 0, 0)

pygame.init()

self.open\_img = pygame.image.load('type-speed-open.png')

self.open\_img = pygame.transform.scale(self.open\_img, (self.w, self.h))

self.bg = pygame.image.load('background.jpg')

self.bg = pygame.transform.scale(self.bg, (900, 600))

self.screen = pygame.display.set\_mode((self.w, self.h))

pygame.display.set\_caption('Type Speed test')

def draw\_text(self, screen, msg, y, fsize, color):

font = pygame.font.Font(None, fsize)

text = font.render(msg, 1, color)

text\_rect = text.get\_rect(center=(self.w / 2, y))

screen.blit(text, text\_rect)

pygame.display.update()

def get\_sentence(self):

global level

f = open('level1.txt').read()

if(level == 1) :

f = open('level1.txt').read()

if (level == 2):

f = open('level2.txt').read()

if (level == 3):

f = open('level3.txt').read()

sentences = f.split('\n')

sentence = random.choice(sentences)

return sentence

def show\_results(self, screen):

global level

if (not self.end):

self.total\_time = time.time() - self.time\_start

count = 0

for i, c in enumerate(self.word):

try:

if self.input\_text[i] == c:

count += 1

except:

pass

self.accuracy = count / len(self.word) \* 100

self.wpm = len(self.input\_text) \* 60 / (5 \* self.total\_time)

self.end = True

print(self.total\_time)

self.results = 'Time:' + str(round(self.total\_time)) + " secs Accuracy:" + str(

round(self.accuracy)) + "%" + ' Wpm: ' + str(round(self.wpm))

self.time\_img = pygame.image.load('icon.png')

self.time\_img = pygame.transform.scale(self.time\_img, (150, 150))

screen.blit(self.time\_img, (self.w / 2 - 75, self.h - 140))

self.draw\_text(screen, "", self.h - 70, 26, (100, 100, 100))

print(self.results)

pygame.display.update()

if (self.accuracy == 100 and self.total\_time < 25):

if (level == 1):

level2()

if (level == 2):

level3()

if (level == 3):

level4()

def run(self):

self.reset\_game()

self.running = True

while (self.running):

clock = pygame.time.Clock()

self.screen.fill((0, 0, 0), (50, 250, 650, 50))

pygame.draw.rect(self.screen, self.HEAD\_C, (50, 250, 650, 50), 2)

self.draw\_text(self.screen, self.input\_text, 274, 26, (250, 250, 250))

pygame.display.update()

for event in pygame.event.get():

if event.type == QUIT:

self.running = False

sys.exit()

elif event.type == pygame.MOUSEBUTTONUP:

x, y = pygame.mouse.get\_pos()

if (x >= 50 and x <= 650 and y >= 250 and y <= 300):

self.active = True

self.input\_text = ''

self.time\_start = time.time()

if (x >= 310 and x <= 510 and y >= 390 and self.end):

self.reset\_game()

x, y = pygame.mouse.get\_pos()

elif event.type == pygame.KEYDOWN:

if self.active and not self.end:

if event.key == pygame.K\_RETURN:

print(self.input\_text)

self.show\_results(self.screen)

print(self.results)

self.draw\_text(self.screen, self.results, 350, 28, self.RESULT\_C)

self.end = True

elif event.key == pygame.K\_BACKSPACE:

self.input\_text = self.input\_text[:-1]

else:

try:

self.input\_text += event.unicode

except:

pass

pygame.display.update()

clock.tick(60)

def reset\_game(self):

self.screen.blit(self.open\_img, (0, 0))

pygame.display.update()

time.sleep(1)

self.reset = False

self.end = False

self.input\_text = ''

self.word = ''

self.time\_start = 0

self.total\_time = 0

self.wpm = 0

self.word = self.get\_sentence()

if (not self.word): self.reset\_game()

# drawing heading

self.screen.fill((0, 0, 0))

self.screen.blit(self.bg, (0, 0))

msg = "Let's test your typing speed !!"

self.draw\_text(self.screen, msg, 60, 50, self.HEAD\_C)

pygame.draw.rect(self.screen, (255, 192, 25), (50, 250, 650, 50), 2)

self.draw\_text(self.screen, self.word, 200, 28, self.TEXT\_C)

pygame.display.update()

Game().run()

Game().run()

def password\_not\_recognised():

global password\_not\_recog\_screen

password\_not\_recog\_screen = Toplevel(login\_screen)

password\_not\_recog\_screen.title("Success")

password\_not\_recog\_screen.geometry("150x100")

Label(password\_not\_recog\_screen, text="Invalid Password ").pack()

Button(password\_not\_recog\_screen, text="OK", command=delete\_password\_not\_recognised).pack()

def user\_not\_found():

global user\_not\_found\_screen

user\_not\_found\_screen = Toplevel(login\_screen)

user\_not\_found\_screen.title("Success")

user\_not\_found\_screen.geometry("150x100")

Label(user\_not\_found\_screen, text="User Not Found").pack()

Button(user\_not\_found\_screen, text="OK", command=delete\_user\_not\_found\_screen).pack()

def delete\_login\_success():

login\_success\_screen.destroy()

def delete\_password\_not\_recognised():

password\_not\_recog\_screen.destroy()

def delete\_user\_not\_found\_screen():

user\_not\_found\_screen.destroy()

def main\_account\_screen():

global main\_screen

main\_screen = Tk()

main\_screen.geometry("300x250")

main\_screen.title("Account Login")

Label(text="Select Your Choice", bg="blue", width="300", height="2", font=("Calibri", 13)).pack()

Label(text="").pack()

Button(text="Login", height="2", width="30", command=login).pack()

Label(text="").pack()

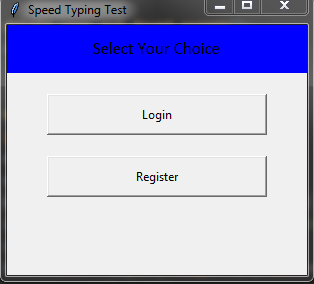
Button(text="Register", height="2", width="30", command=register).pack()

main\_screen.mainloop()

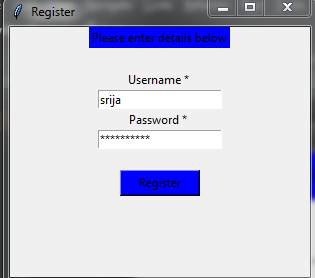
main\_account\_screen()

**RESULTS:**

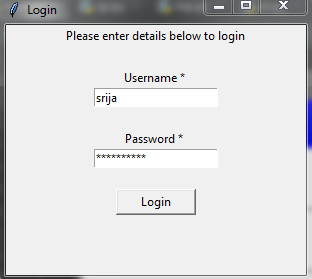
**Execution:**

****

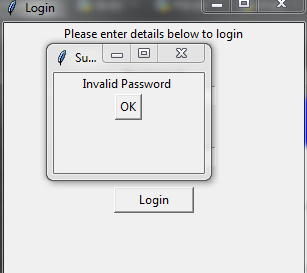
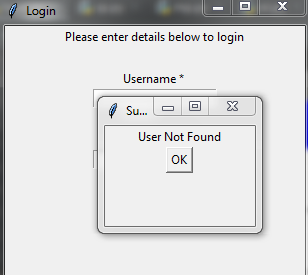
**Registration Page:**

****

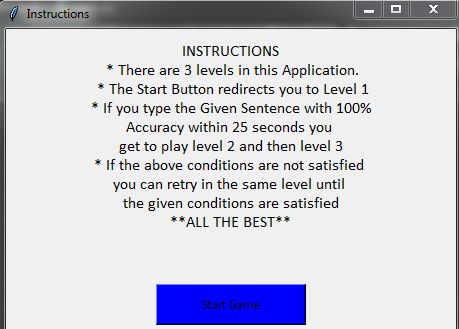
**Login Page:**

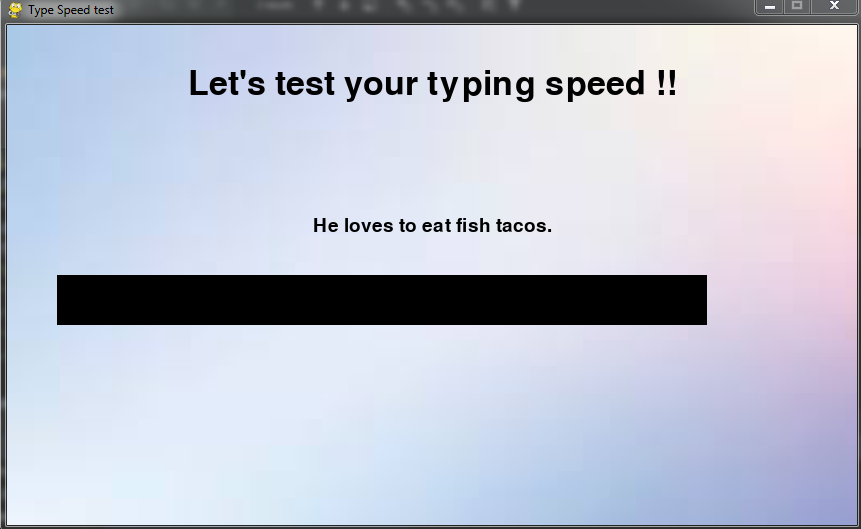
****

**Login errors:**

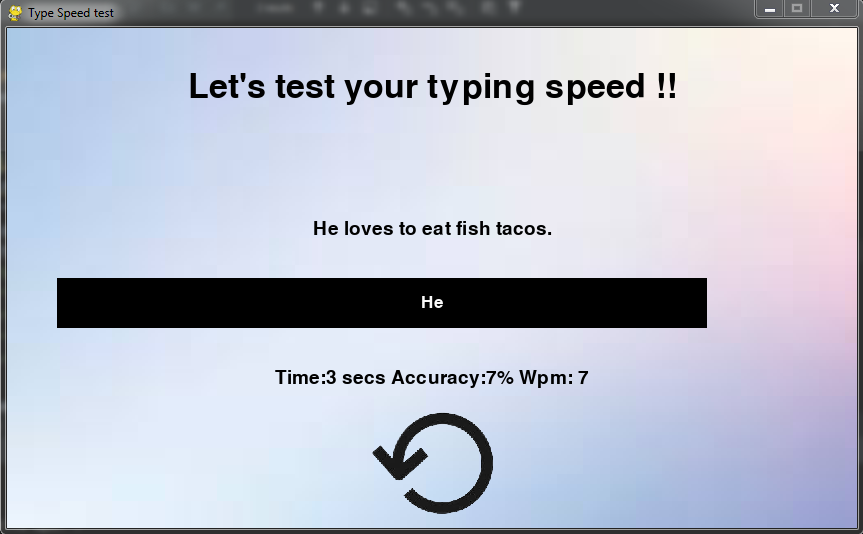
** **

**Instructions:**

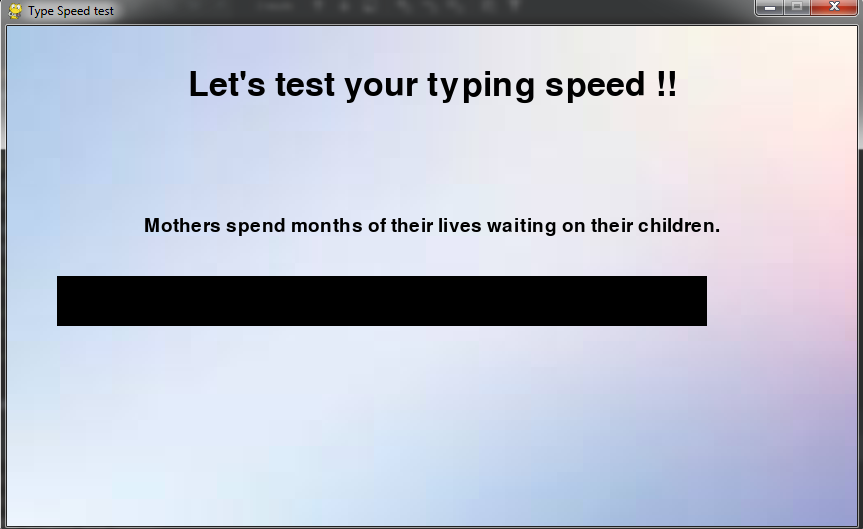
****

**Level 1 :**

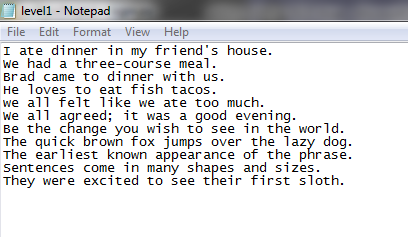
**Level Retry:**

****

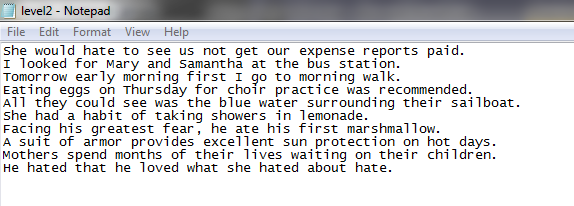
**Next Levels:**

****

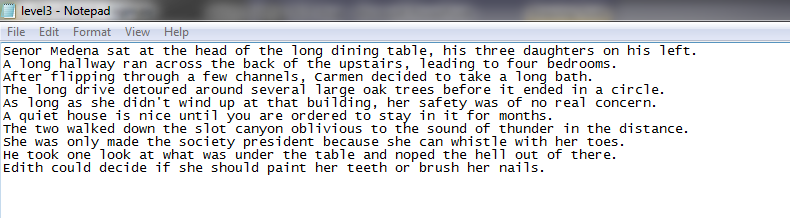
**Sentences of Level 1:**

****

**Sentences of Level 2:**

****

**Sentences of Level 3:**

****

**ADDITIONAL KNOWLEDGE ACQUIRED**

* Python GUI programming using the module tkinter .
* How to use different modules in python.
* Use case diagrams.
* Activity diagram.

**CONCLUSION**

* Speed typing test contains 3 levels.
* The game starts with level1, if the user types the sentence with 100% accuracy within 25 sec then it automatically goes to next level and so on.
* If it does not reach target same level will be repeated.
* Moving to each level the size of the sentences will be increased.

**FUTURE WORK**

* We would like to add more features to our project like calculating the high score and also display the previous performances and also a time limit is given to the user to type in the sentence.
* We would like to add more levels to the game and the user can continue from which level they stopped.

**REFERENCES**

Basic Python

1. Course covered during 2nd semester by Keizia rani madam
2. Ppts and handouts provided by the madam.
3. Python Programming - Using Problem Solving Approach First Edition by Reema Thareja.

Use cases and activity diagram

1. <https://www.youtube.com/watch?v=zid-MVo7M-E>
2. <https://www.youtube.com/watch?v=knM8BGY9yVI&t=161s>