Configuration Manual

MSc Research Project
CyberSecurity

Anushka Ingale
Student ID: x20125020

School of Computing
National College of Ireland

Supervisor: Niall Heffernan

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Anushka Ingale
Student ID:	x20125020
Programme:	CyberSecurity
Year:	2021
Module:	MSc Research Project
Supervisor:	Niall Heffernan
Submission Due Date:	23/09/2021
Project Title:	Configuration Manual
Word Count:	XXX
Page Count:	8

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	22nd September 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Anushka Ingale
x20125020

1 Introduction

This document is divided into subject sections to make it easier to find the information needed to format the project and to instruct the user on how to rebuild the project and complete it successfully. The system is implemented using the method presented in the report, and the configuration details are delivered in this document to aid with project comprehension, setting up the environment, and execution. Python is the programming language used in this project.

2 Hardware specification

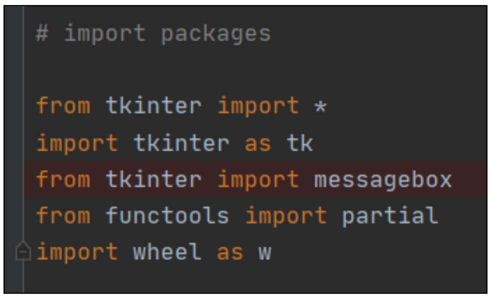
- Device Name - MSI
- Processor - Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.60 GHz
- RAM - 16 GB
- ROM - 4 GB
- OS - Windows
- System type 64-bit operating system, x64-based processor

Software requirements

Programming language: Python 3.9.6
Python IDE: Pycharm 2020.2.2

3 Libraries invoked

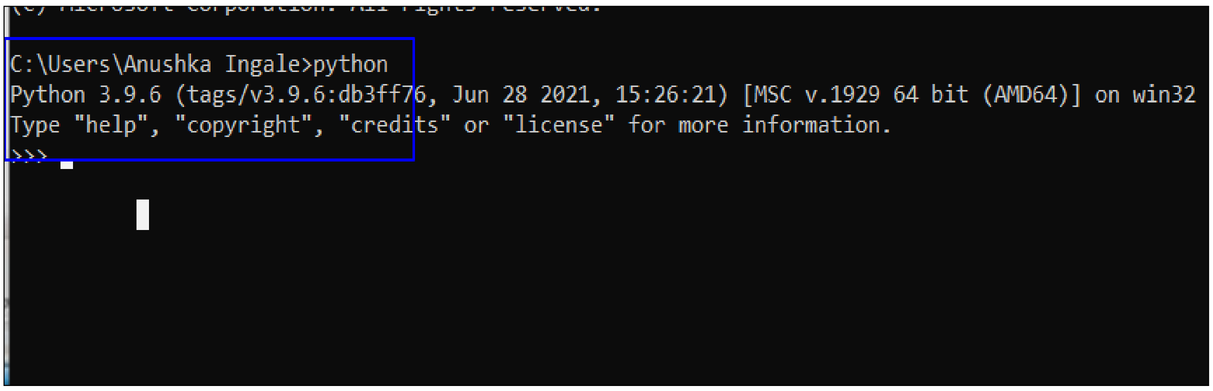
PIP for python 3 - sudo apt-get install python3. Figure 1 Figure 2



```
# import packages

from tkinter import *
import tkinter as tk
from tkinter import messagebox
from functools import partial
import wheel as w
```

Figure 1: Importing Libraries



```
C:\Users\Anushka Ingale>python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 2: Install Python

4 Implementation

1) Pycharm IDE is necessary to build a convenient and productive interface in the Python language, and this study employs pycharm version 2020.2.2. We download the Community edition from the JetBrains website and install it. Figure 3

2) Open the Rubic folder in pycharm. To open the file, use the following path, and the folder location must be accessed and opened as shown below.Figure 4

3) Once the project has been opened, use the run option from the title bar to run it, as seen below. Figure 5 Figure 6

4) The Rubic window displays, with two options: login and registration. Clicking on a specific button will invoke the function, which will run it and display the page.

5) During the registration step, the user must select a username and password while keeping in mind the character restriction, and then submit it to save the information into the system. The data is stored as a text file in the same Rubic folder. Figure 7

6) The image below depicts the login screen, which requires user input, such as the username and password, which should be typed using specific keys.

7) The wheel spins clockwise and anticlockwise when specific keys are entered, as indicated in the figure. The purpose and rationale for wheel rotation are also depicted in the same. Figure 8 Figure 9

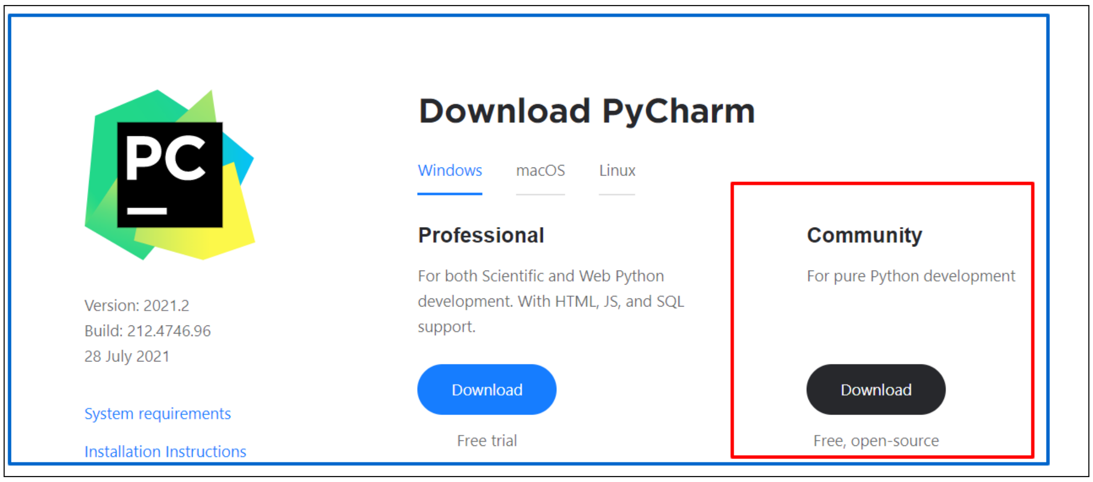


Figure 3: Download Pycharm

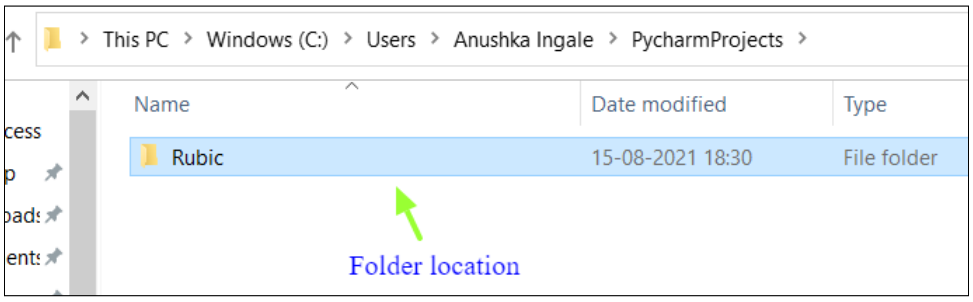


Figure 4: Location of Rubic folder

i) c = for clockwise rotation of the wheel
ii) a = anticlockwise rotation of the wheel
iii) i= for selecting a letter/number from the inner sector
iv) o = for selecting a letter/number from the outer sector

8) The user must spin the wheel using the color he selected at registration to reach the first letter or number required for password entry.

9) Additional special keys ”i” and ”o” must be used to pick letters in relation to the wheel’s inner and outer orbits. Figure 10

10) As seen below, the entered password has a “*” value for password privacy. Figure 11

11) Entering correct credentials will result in a successful login, whereas entering invalid credentials will result in a failed login. Figure 12 Figure 13

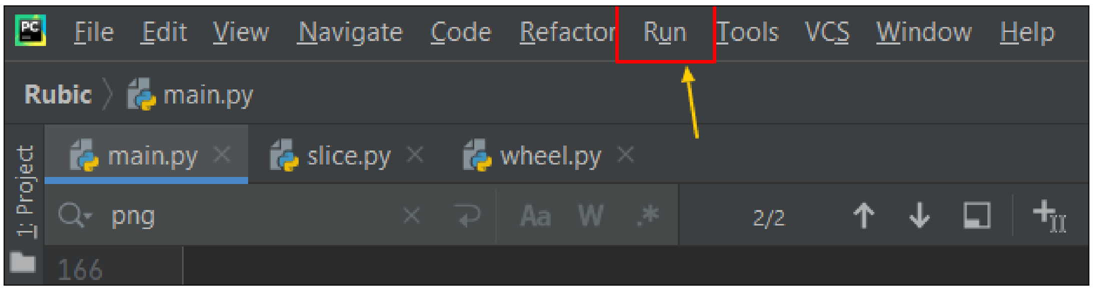


Figure 5: Running file

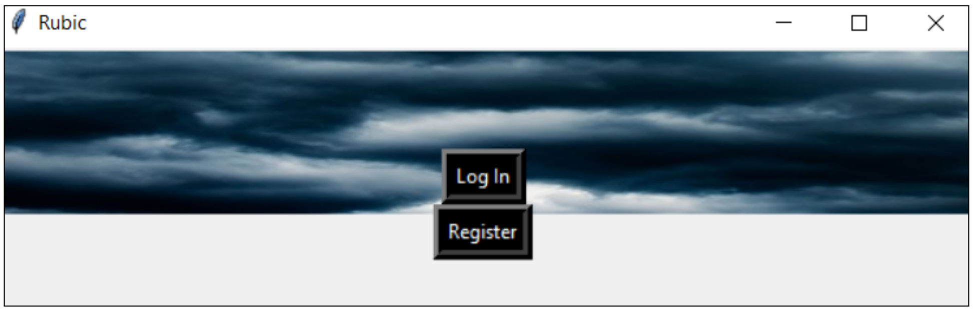
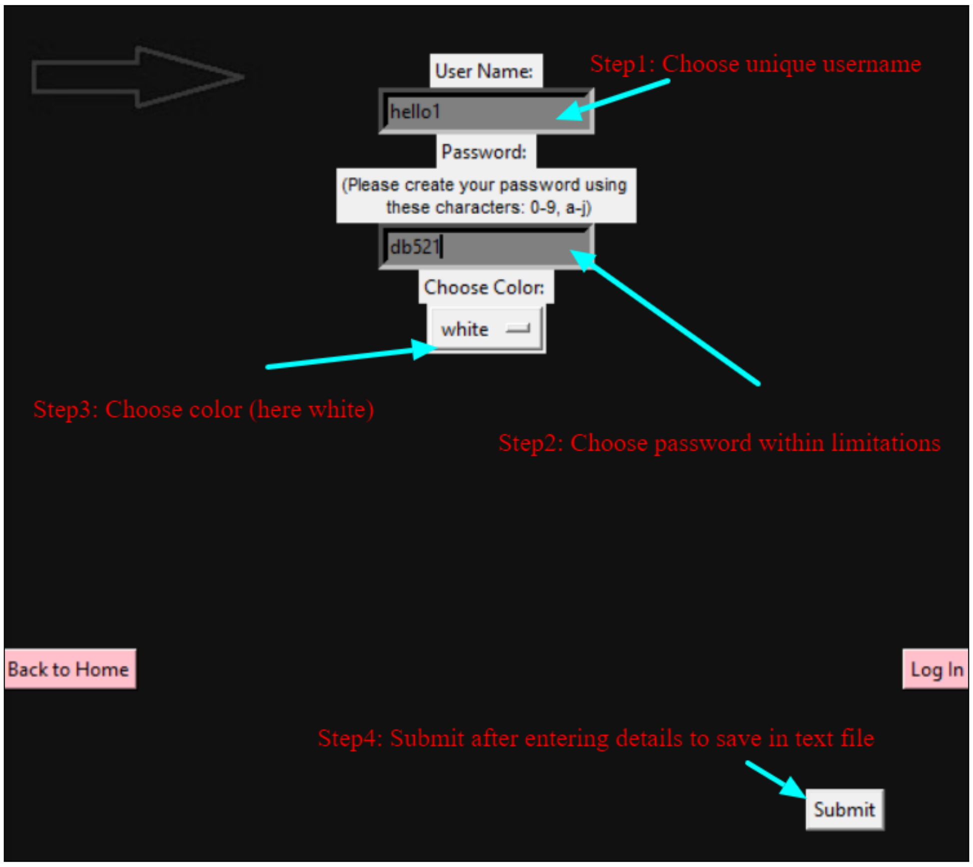


Figure 6: Rubic window



Step1: Choose unique username

Step3: Choose color (here white)

Step2: Choose password within limitations

Step4: Submit after entering details to save in text file

Figure 7: Registration steps

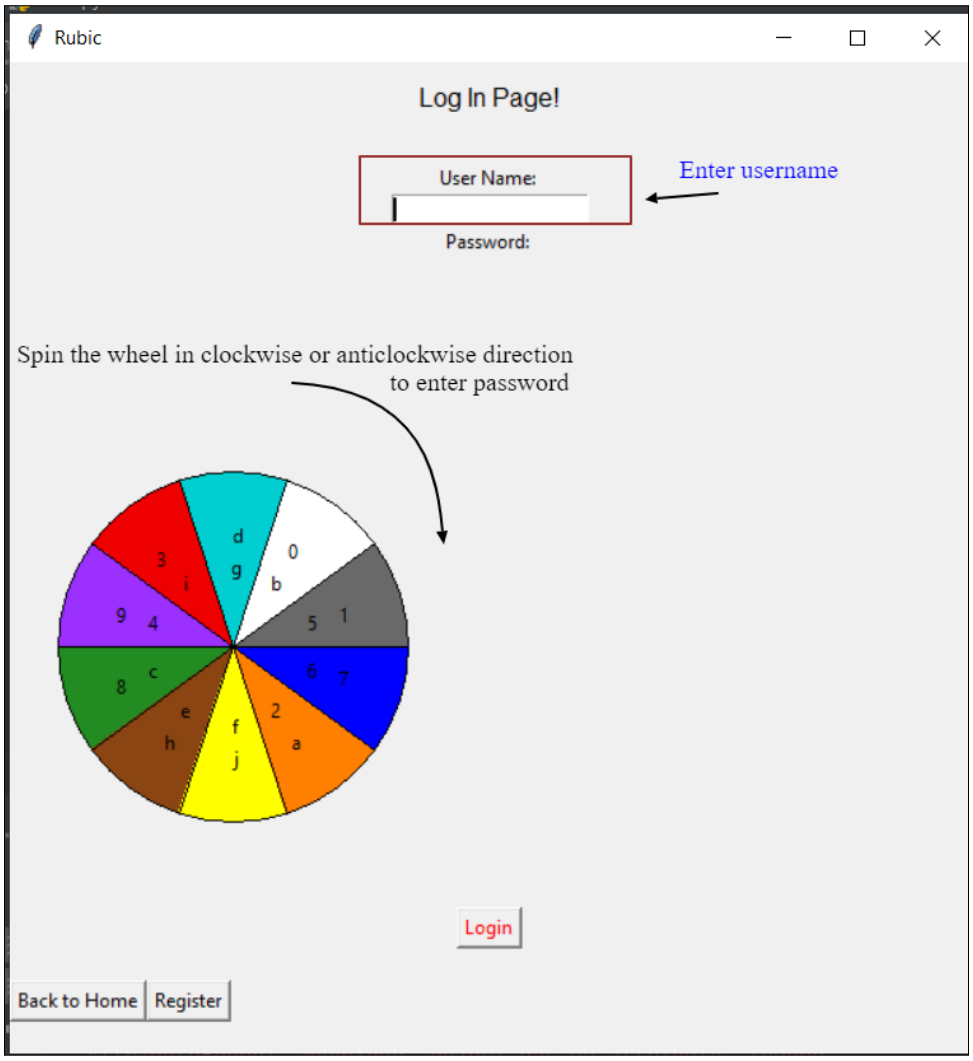
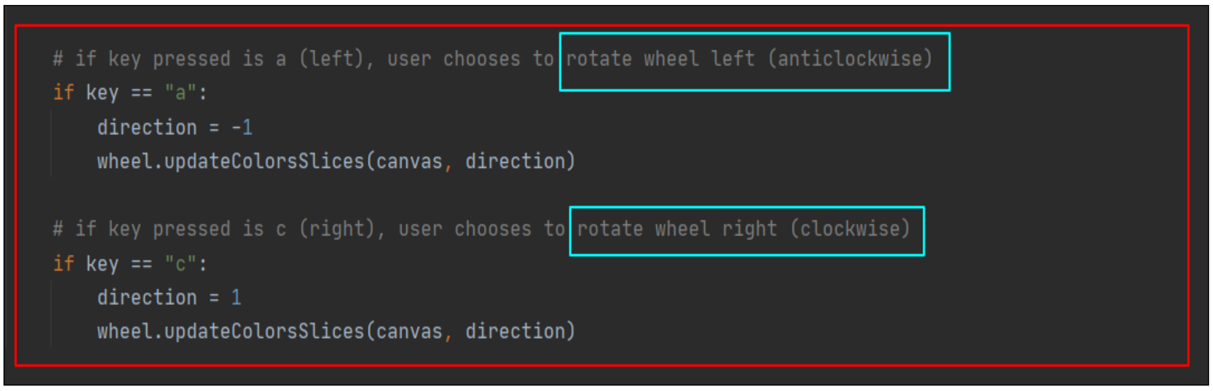


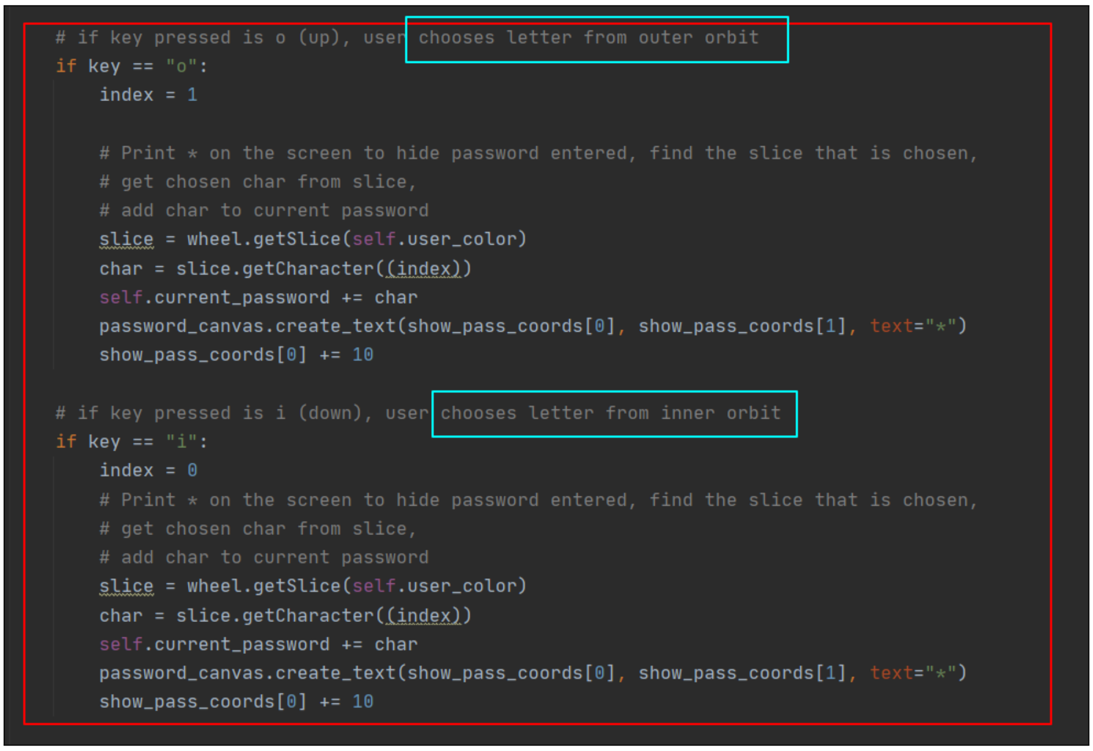
Figure 8: Login Page



```
# if key pressed is a (left), user chooses to rotate wheel left (anticlockwise)
if key == "a":
    direction = -1
    wheel.updateColorsSlices(canvas, direction)

# if key pressed is c (right), user chooses to rotate wheel right (clockwise)
if key == "c":
    direction = 1
    wheel.updateColorsSlices(canvas, direction)
```

Figure 9: Logic for rotation of wheel

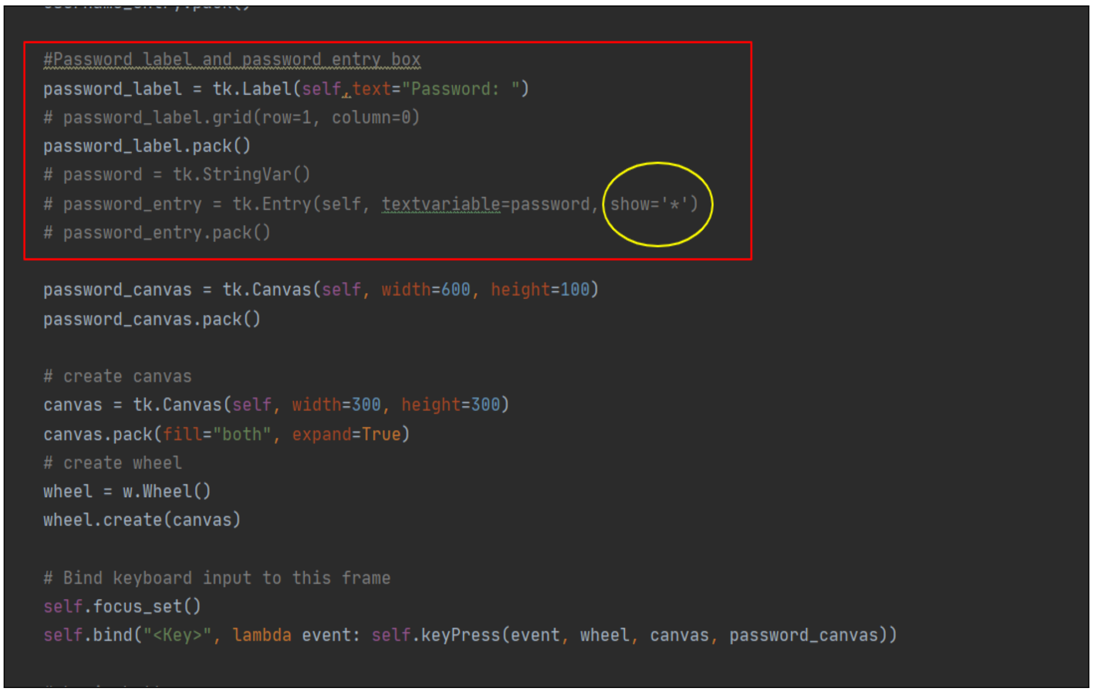


```
# if key pressed is o (up), user chooses letter from outer orbit
if key == "o":
    index = 1

    # Print * on the screen to hide password entered, find the slice that is chosen,
    # get chosen char from slice,
    # add char to current password
    slice = wheel.getSlice(self.user_color)
    char = slice.getCharacter((index))
    self.current_password += char
    password_canvas.create_text(show_pass_coords[0], show_pass_coords[1], text="*")
    show_pass_coords[0] += 10

# if key pressed is i (down), user chooses letter from inner orbit
if key == "i":
    index = 0
    # Print * on the screen to hide password entered, find the slice that is chosen,
    # get chosen char from slice,
    # add char to current password
    slice = wheel.getSlice(self.user_color)
    char = slice.getCharacter((index))
    self.current_password += char
    password_canvas.create_text(show_pass_coords[0], show_pass_coords[1], text="*")
    show_pass_coords[0] += 10
```

Figure 10: Logic for special keys



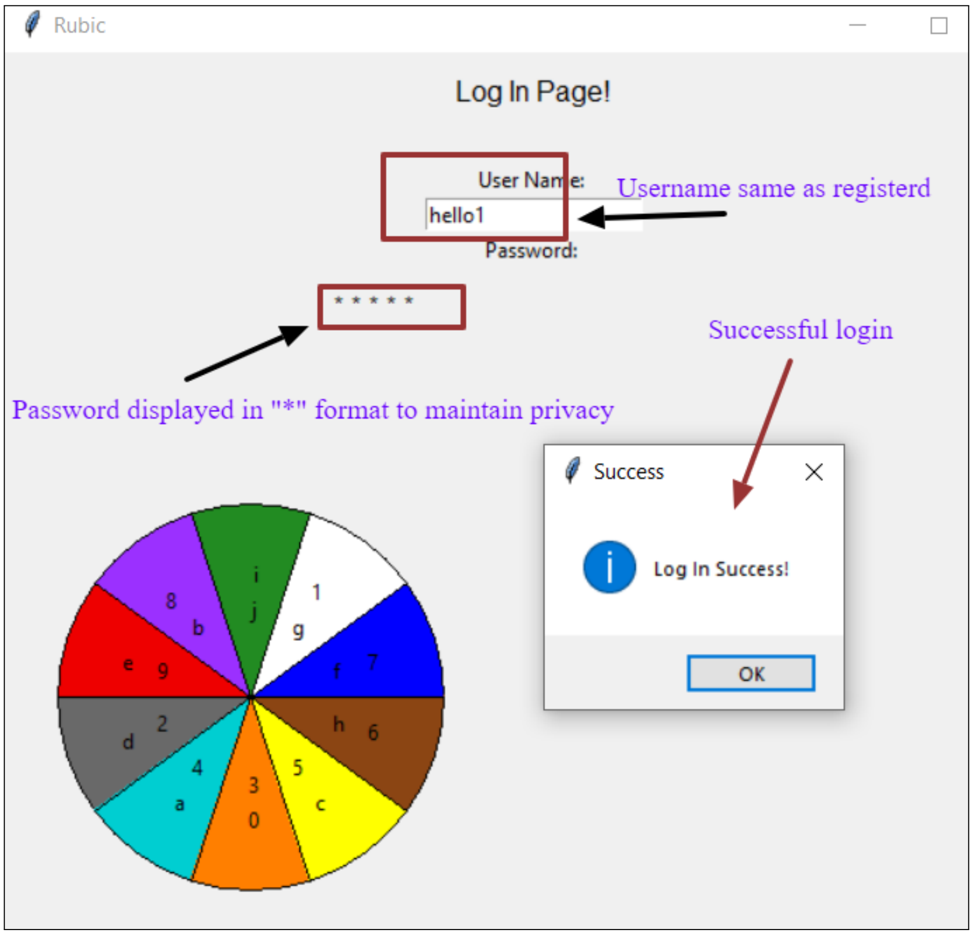
```
#Password label and password entry box
password_label = tk.Label(self, text="Password: ")
# password_label.grid(row=1, column=0)
password_label.pack()
# password = tk.StringVar()
# password_entry = tk.Entry(self, textvariable=password, show='*')
# password_entry.pack()

password_canvas = tk.Canvas(self, width=600, height=100)
password_canvas.pack()

# create canvas
canvas = tk.Canvas(self, width=300, height=300)
canvas.pack(fill="both", expand=True)
# create wheel
wheel = w.Wheel()
wheel.create(canvas)

# Bind keyboard input to this frame
self.focus_set()
self.bind("<Key>", lambda event: self.keyPress(event, wheel, canvas, password_canvas))
```

Figure 11: Secure password entry


Rubic

Log In Page!

User Name: hello1

Password: *****

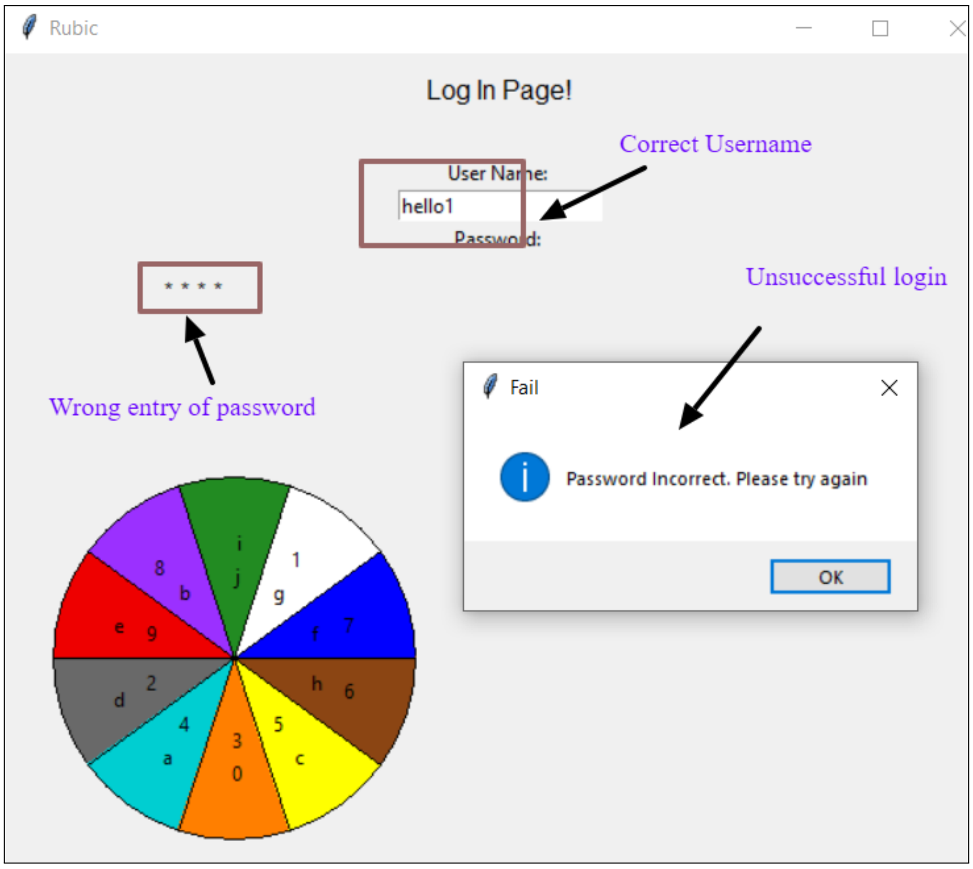
Success

Log In Success!

OK

Username same as registerd
Password displayed in "*" format to maintain privacy
Successful login
i
j
1
g
8
b
e
9
f
7
d
2
h
6
4
a
3
0
5
c

Figure 12: Successful login



Password Incorrect. Please try again

Figure 13: Invalid login credentials