

WIZARD: Secure Marketing Application Project – Design, Development & Testing

The following program has been named WIZARD. This new application contains the necessary features to host promotions within its own special separate category combined with generating and scanning QR codes. These QR codes embed extra information relating to the promotions. The graphical user interface of the software is simplistic yet decently appealing for users to interact with as they use the application. It provides basic user authentication and validation such as password hashing to prevent SQL injections and prevent minors from accessing WIZARD when a user signs up for a WIZARD account.

Design & Implementation of the WIZARD Application

All phases carried out within the WIZARD project followed the “Waterfall Model” (Royce, 1970). It was applied as an approach to design, implement and test the WIZARD application. During the design phase, several flowcharts and an entity relationship diagram were produced to represent the necessary logical processes and requirements involved in the application. Additionally, minimal knowledge on art design was needed to present WIZARD windows as user-friendly and easy to navigate for users. At the end, the length of the design phase was on schedule and all works accomplished with no issues.

Before the development of the WIZARD secure marketing application, it was required to apply the basic principles of Python programming such as functions and sequence statements, especially applying GUI. Tkinter was used however the extension customtkinter was heavily used as it is more popular and it allows widget customisation to make the GUI more aesthetically pleasing (CustomTkinter Documentation, 2021). Within the source code, to implement certain features like password hashing it was required to add Python libraries and modules into the source code. One Python library that was used was Sqlite3. It is needed to connect the SQLite database to the Python source code in order to carry out data processing (Python Documentation, 2023).

```
1      #Imported libraries and modules will be located here
2
3      import tkinter
4      from tkinter import messagebox, PhotoImage
5      import customtkinter
6      import bcrypt
7      import sqlite3
8      import qrcode
9      import cv2
10     import image
11
```

Figure 1 – Imported Python modules and libraries

Structure of the WIZARD Application

Signup Window – This window enables users to have the authentication to login into WIZARD by inputting necessary credentials. These credentials are validated such as their age to avoid users under 18 having access to the application along with their email having to contain an “@gmail” or a message box error would pop up. They are stored within the database and a new table called “USERS” is produced to store the data. Additional authentication includes hashing the user’s password and requiring the password length to be more than 10 characters. When all requirements are followed where filling all credentials is one of them, a message box is prompted that a WIZARD account has been created. This means that the data is successfully stored within the database.

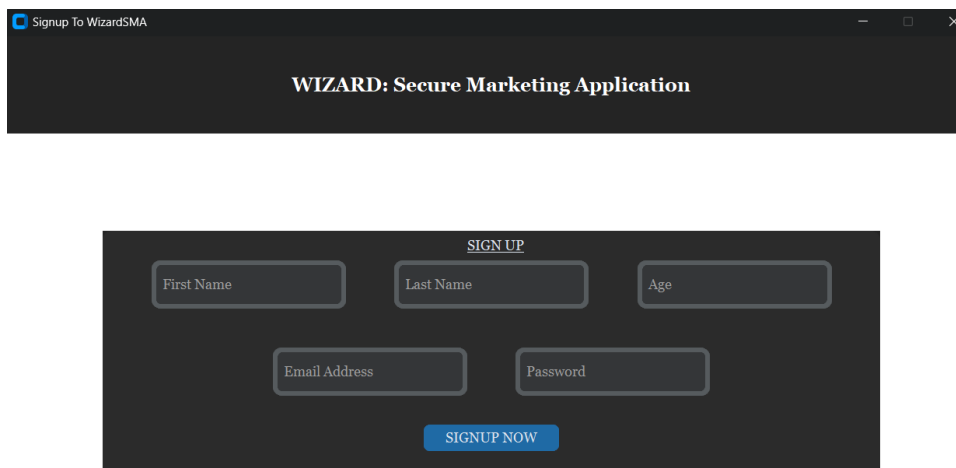


Figure 2 – WIZARD Signup Window

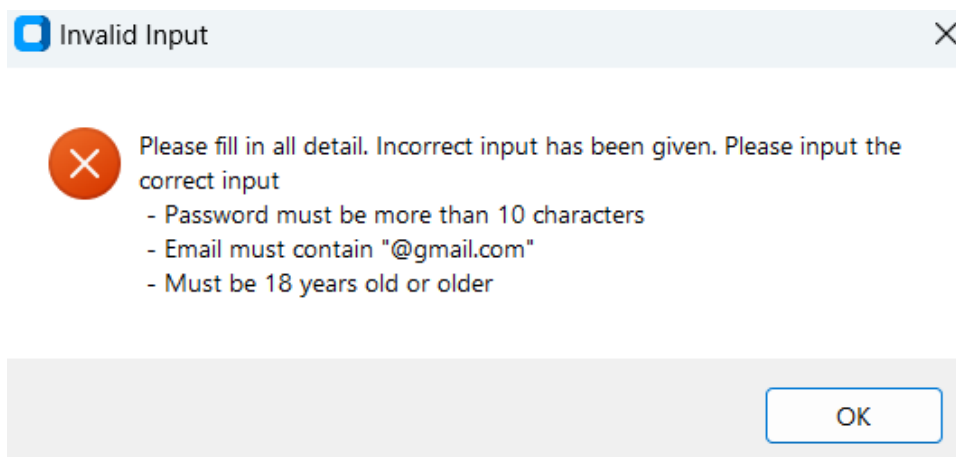


Figure 3 – Error Message Box when input is invalid

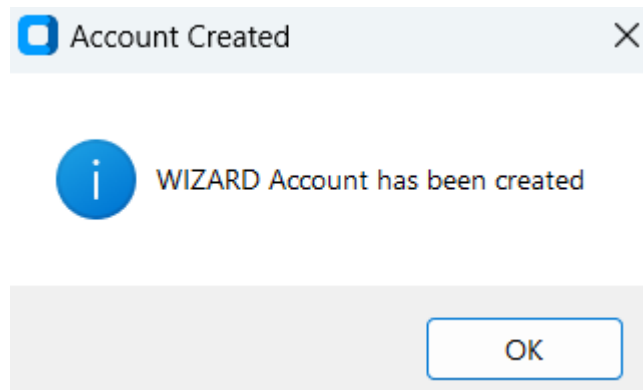


Figure 4 – Information Message Box when WIZARD account created

```

71 def sign_up():
72     requirement_1 = "@gmail.com"
73     add_to_database = sqlite3.connect("WIZARD.db")
74     connected_database = add_to_database.cursor()
75     connected_database.execute('CREATE TABLE IF NOT EXISTS USERS (
76         User_ID INTEGER PRIMARY KEY AUTOINCREMENT,
77         First_Name TEXT NOT NULL,
78         Last_Name TEXT NOT NULL,
79         Age INTEGER NOT NULL,
80         Email_Address TEXT NOT NULL,
81         Password TEXT NOT NULL)')
82
83     first_name = first.get()
84     last_name = last.get()
85     a = int(age.get())
86     mail = email_address.get()
87     en_code_password = password_signup.get()
88
89     if first_name != "" and last_name != "" and a != "" and mail != "" and en_code_password != "" and a >= 18 and mail.__contains__(requirement_1) and len(en_code_p
90         hashed_password = bcrypt.hashpw(en_code_password.encode('utf-8'), bcrypt.gensalt())
91         connected_database.execute('INSERT INTO USERS (First_Name, Last_Name, Age, Email_Address, Password) VALUES (?, ?, ?, ?, ?)', _parameters: [first_name, last_na
92         add_to_database.commit()
93         messagebox.showinfo( "Account Created", message: "WIZARD Account has been created")
94         add_to_database.close()
95     else:
96         messagebox.showerror( "Invalid Input", message: "Please fill in all detail. Incorrect input has been given. Please input the correct input\n - Password mu

```

Figure 5 – Python code for signing up

Data inputted are the first name, last name, age email address and password of the user into five GUI entry widgets with placeholders to provide simple guidance along with a button to allow user to prompt account creation. The window has a minimal appearance with two entry widget and two buttons, lacking attractive appearance but follows required functionalities to allow users to input details, stores those details, hash the user's password and click the signup button to be notified to have a WIZARD account. However, there could've been more to have considered and implemented when validating the credentials given. For example, instead of only applying the constraint for the length of the password, the user's password could be checked to follow all the universal rules for secure passwords like containing numbers, symbols and uppercase letters. Another change could be added is the validation of the email address. The constraint for the email to only contain "@gmail" causes limitation. This is due to lack of knowledge on validating effectively using Python.

```

44 def to_signup():
45     signup_window = customtkinter.CTkToplevel()
46     signup_window.title("Signup To WizardSMA")
47     signup_window.geometry("1000x550")
48     signup_window.config(bg="white")
49     signup_window.resizable( width=0, height=0)
50
51     global first,last,age_email_address,password_signup
52     customtkinter.CTkLabel(master=signup_window, width=1000, height=100, text="WIZARD: Secure Marketing Application", text_color="white", font=font_1).pack(anchor=tkinter.CENTER)
53     signup_frame = customtkinter.CTkFrame(master=signup_window, width=1000, height=825, corner_radius=10)
54     signup_frame.pack(padx=100, pady=100)
55     customtkinter.CTkLabel(master=signup_frame, text="SIGN UP", font=font_4).place(x=375, y=0.5)
56     first = customtkinter.CTkEntry(master=signup_frame, placeholder_text="First Name", font=font_2, width=200, height=50, border_width=5, corner_radius=10)
57     first.place(x=50, y=30)
58     last = customtkinter.CTkEntry(master=signup_frame, placeholder_text="Last Name", font=font_2, width=200, height=50, border_width=5, corner_radius=10)
59     last.place(x=300, y=30)
60     age = customtkinter.CTkEntry(master=signup_frame, placeholder_text="Age", font=font_2, width=200, height=50, border_width=5, corner_radius=10)
61     age.place(x=550, y=30)
62     email_address = customtkinter.CTkEntry(master=signup_frame, placeholder_text="Email Address", font=font_2, width=200, height=50, border_width=5, corner_radius=10)
63     email_address.place(x=175, y=120)
64     password_signup = customtkinter.CTkEntry(master=signup_frame, placeholder_text="Password", font=font_2, width=200, height=50, border_width=5, corner_radius=10, show="*")
65     password_signup.place(x=425, y=120)
66
67     button = customtkinter.CTkButton(master=signup_frame, text="SIGNUP NOW", font=font_2, hover_color="grey", command=sign_up)
68     button.place(relx=0.5, rely=0.85, anchor=tkinter.CENTER)
69
70

```

Figure 6 – Python code to show signup window GUI

Login Window – WIZARD must have a login window to provide basic user authentication to access the application using data stored within the database. Once a user has created an account through the signup window, they should be able to input their email address and password into the two GUI entry widgets with placeholders, providing simple guidance for users along with a button to allow user to log in. Another functionality included is the ability to open the signup window if it is the user's first-time using WIZARD, wishing to access it by storing their details in the database using a button to prompt this process.

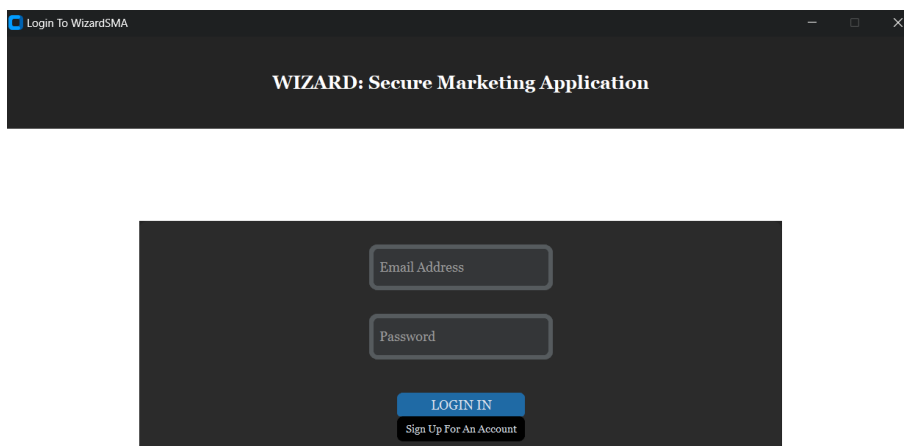


Figure 7 – WIZARD Login Window

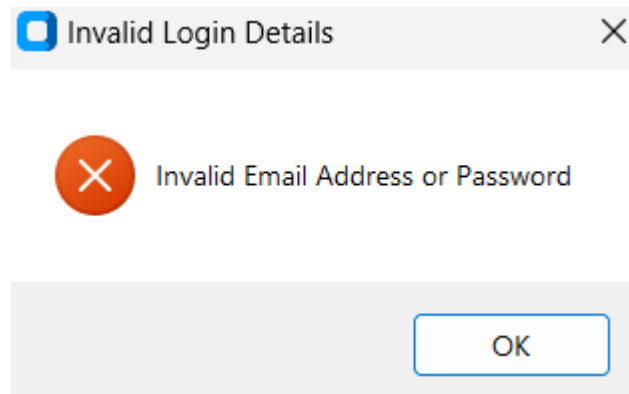


Figure 8 – Error Message Box when input is invalid at login window

It follows the necessary functionalities needed for the user to login.

```

6 def login():
7     e_address = e_address_textbox.get()
8     password_login = password_textbox.get()
9
10    if e_address == "" or password_login == "":
11        messagebox.showerror( title: "Login Error", message: "Please input missing details")
12
13    else:
14        add_to_database = sqlite3.connect("WIZARD.db")
15        connect_to_database = add_to_database.cursor()
16        connect_to_database.execute('''CREATE TABLE IF NOT EXISTS USERS (
17                                     User_ID INTEGER PRIMARY KEY AUTOINCREMENT,
18                                     First_Name TEXT NOT NULL,
19                                     Last_Name TEXT NOT NULL,
20                                     Age INTEGER NOT NULL,
21                                     Email_Address TEXT NOT NULL,
22                                     Password TEXT NOT NULL)''')
23
24        connect_to_database.execute( __sql: 'SELECT Password FROM USERS WHERE Email_Address=?', __parameters: [e_address,])
25        result = connect_to_database.fetchone()
26
27        if result and bcrypt.checkpw(password_login.encode('utf-8'),result[0]):
28            messagebox.showinfo( title: "Login Successful", message: "You have successfully logged in")
29            promotion_page()
30
31        else:
32            messagebox.showerror( title: "Invalid Login Details", message: "Invalid Email Address or Password")

```

Figure 9 – Python Code for Login window

Promotions & QRCode Window - The main purpose of the WIZARD application is to host the promotions. This window specifically contains the function to generate and scan QRCode associated with the promotions. As soon as users log into the application, a new window is opened that contains a tab view that can switch between the “Promotions” GUI and the “QR Scanner” when triggered. The “Promotions” tab contains four frames represented as promotions indicated by their colours. Each promotion contains a button that generates a QRcode with data containing extra details on that promotion. The generated QRcodes are stored within a folder. The “QR Scanner” tab provides a frame combined with an entry and button. The image path of the QRCode should be inputted into the entry. The button is clicked and the QRcode is scanned, revealing the data as a label in the frame.

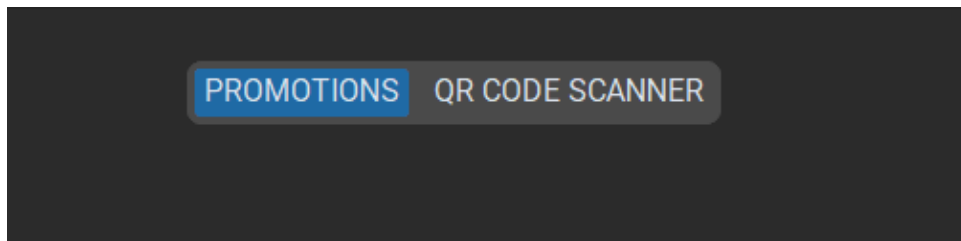


Figure 10 – TabView window

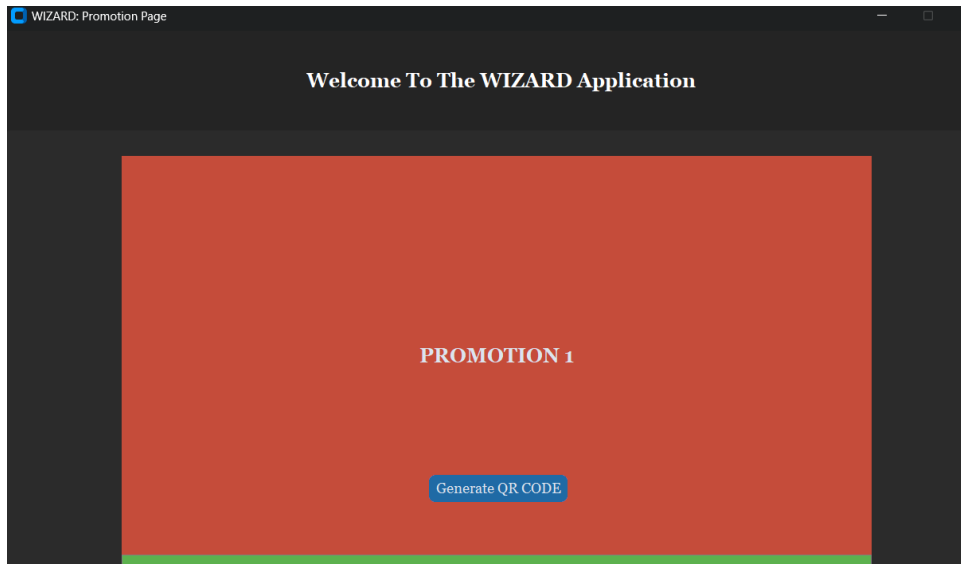


Figure 11 – Promotions Tab in TabView window

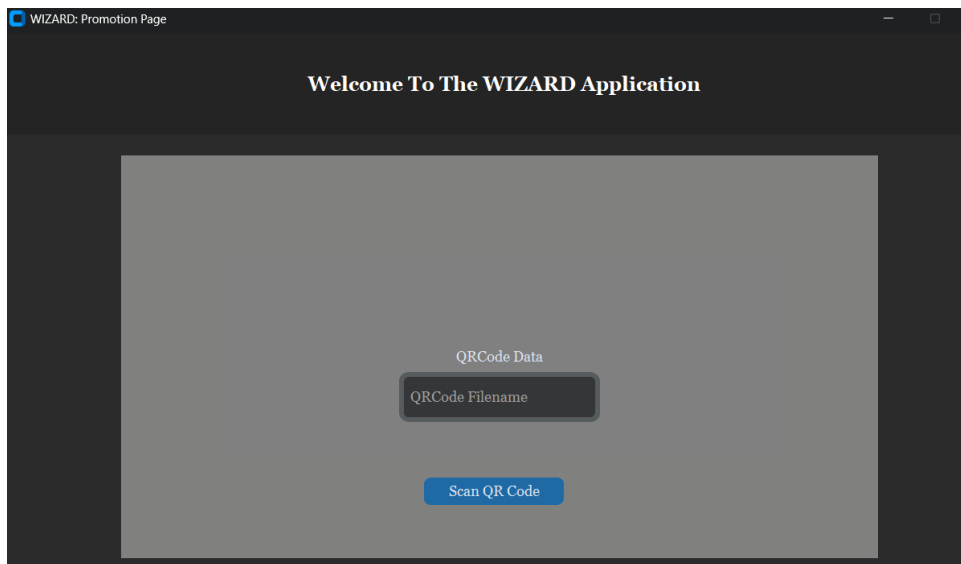


Figure 11 – QRCode Scanner Tab in TabView window

Testing

Test Case for Signup Window:

Test Case Type	Description	Step in test	Expected Result	Status
Signup Window Functionality	When all details are inputted, the details are stored in the database. Password is hashed and an information message box is prompt	Input all details correctly and click the "SIGNUP NOW" window	An information message box is prompt to indicate account creation. Password is hashed and stored in database	Pass
Signup Window Validation	When the length of the password is less than 10 characters, error message box appears	All details inputted but length of password is less than 8 characters	An error message box is prompt to indicate invalid data. Data is not stored in database	Pass
Signup Window Validation	When the age inputted is less than 18, error message box appears	All details inputted but age inputted is less than 18	An error message box is prompt to indicate invalid data. Data is not stored in database	Pass
Signup Window Validation	If all details are not filled in, error message box appears	Some details are not inputted	An error message box is prompt to indicate invalid data. Data is not stored in database	Pass
Signup Window Validation	Details are not filled in, error message box appears	No details are not inputted	An error message box is prompt to indicate invalid data. Data is not stored in database	Fail

Test Case for Login Window:

Test Case Type	Description	Step in test	Expected Result	Status
Login Window Functionality	When correct email address and password	Input proper email address and password	An information message box is prompt to	Pass

	inputted, the "Promotions and QRCode" window should be opened		indicate login and "Promotions and QRCode" window is created	
Login Window Functionality	When "Sign up for an account" button is pressed, the signup window should appear	Click the "Sign up for an account" button	The signup window is opened	Pass
Login Window Input Authentication	If nothing is inputted into the entries, an error message box is prompt	Click the "LOGIN" button without inputting email address and password	An error message box is prompt to input details	Pass
Login Window Input Authentication	If only one of the entries are inputted, an error message box is prompt	Click the "LOGIN" button with one of the entries inputted	An error message box is prompt to input details	Pass
Login Window Input Authentication	When the email address or password is inputted incorrectly, an error message box is prompt	Click the "LOGIN" button with incorrect details	An error message box is prompt that details are invalid	Pass

Test Case for Promotion & QRCode Window:

Test Case Type	Description	Step in test	Expected Result	Status
Promotion and QRCode Window Functionality	When the "Generate QR Code" button is clicked, a QRcode relating to the promotion is generated in a folder	Click "Generate QR Code" button	A QRCode is generated into a folder and an information message box is shown to indicate generation	Pass
Promotion and QRCode Window Functionality	When the "Scan QR Code" button is clicked, the QRcode should be scanned	Filename of QRCode is inputted into entry and "Scan QR Code" button is clicked	QRCode is scanned, showing QRcode data	Fail

Promotion and QRCode Window Validation	If something else is inputted into the entry, an error message box is prompt	Click "Scan QR Code" button with random letters in entry	An error message box is prompt invalid filename for QRCode	Fail
--	--	--	--	------

The "Promotions and QRCode Window" test case contained failed tests relating to storing data in the database with the promotions and failing to scan the qrcode and display the data on the label.

In conclusion, the WIZARD application had a limited scope due to the development phase being behind schedule however it follows the basic main function required. The lack of eye-catching design which all the windows shares are due to lack of knowledge on customtkinter however it is essential that this changes as WIZARD is frequently updated.

Reference List

- Royce, W.W. (1970) 'Managing the development of Large Software Systems (1970)', *Ideas That Created the Future*, pp. 321–332. doi:10.7551/mitpress/12274.003.0035.
- *Python 3.12.1 documentation* (2023) *3.12.1 Documentation*. Available at: <https://docs.python.org/3/> (Accessed: 08 January 2024).
- *Customtkinter* (2021) *PyPI*. Available at: <https://pypi.org/project/customtkinter/0.3/> (Accessed: 08 January 2024).
-