Project Report on

STOCK MANAGEMENT PROGRAM

Submitted to

JAYSHREE PERIWAL HIGH SCHOOL

3, Chitrakoot Scheme

Jaipur

In partial fulfillment

of the requirements for

All India Senior School Certificate Examination 2025

Of

CENTRAL BOARD OF SECONDARY EDUCATION

Submitted by:

Aditya Goyal (Roll No.)

Shreyas Nair (Roll No.)

ACKNOWLEDGEMENT

We would like to thank everyone who helped us to accomplish this project.

Our sincere thanks to ***respected teachers*** who have helped us with their valuable suggestions and support throughout the development of the project.

We would like to thank our project guide ***Ms. Himanshi Sharma*** for providing guidance and support at every stage of the project.

We are extremely grateful to **Mrs. Jayshree Periwal, Director** and ***Mrs. Madhu Maini***, Principal of ***JAYSHREE PERIWAL HIGH SCHOOL, JAIPUR***, for providing us with a computer lab, due to which we were able to complete our project.

Aditya Goyal

Shreyas Nair

CERTIFICATE OF ORIGINALITY

This is to certify that the project entitled "**Stock Management Program**" submitted to ***JAYSHREE PERIWAL HIGH SCHOOL*** in partial fulfillment of the requirement for ***All India Senior School Certificate Examination (AISSCE*)** **2025** of **CBSE**, is original work carried out by ***Aditya Goyal***, ***Shreyas Nair*** under my guidance.

The matter embodied in this project is genuine work done by the students and has not been submitted by any course of study.

_____

Signature of the guide

Date: _____

Name: **Ms. Himanshi Sharma**

HOD, Computer Science

JAYSHREE PERIWAL HIGH SCHOOL

Jaipur

CONTENTS

1. Objective & Scope of the Project

Objective

The main objective of the project is to act as Tool to keep track of your Stock Investments.

This package is useful for checking latest Stock Prices and can even act as a Mock Trading.

Scope

This project was developed as a part of XII Standard Course.

Further, it can be used by any adult or child to learn more about the ups and downs of the Stock Market by analyzing Real Time Data.

2. Problem Definition

The Project “Stock Management System” allows the user to search up any Stock in the Stock Exchange and analyze the history of its Stock Prices and even use it as an application for Mock Trading.

The application beautifully depicts the Stock Prices using a Line Graph and also displays various details about the Company such as Website Link, Business Summary etc.

3. Life Cycle of the Project

System Development Life Cycle (SDLC)

The System Development Life Cycle (SDLC) is a set of activities that analysts, designers and users carry out to develop and implement an Information System.

The SDLC consists of the following activities.

Feasibility Study

↓

Requirement Definition (Analysis)

↓

Design (Database & Program)

↓

Development of Software

↓

Unit Testing

↓

System Testing

↓

Implementation

↓

Evaluation

↓

Maintenance

4. System Specifications

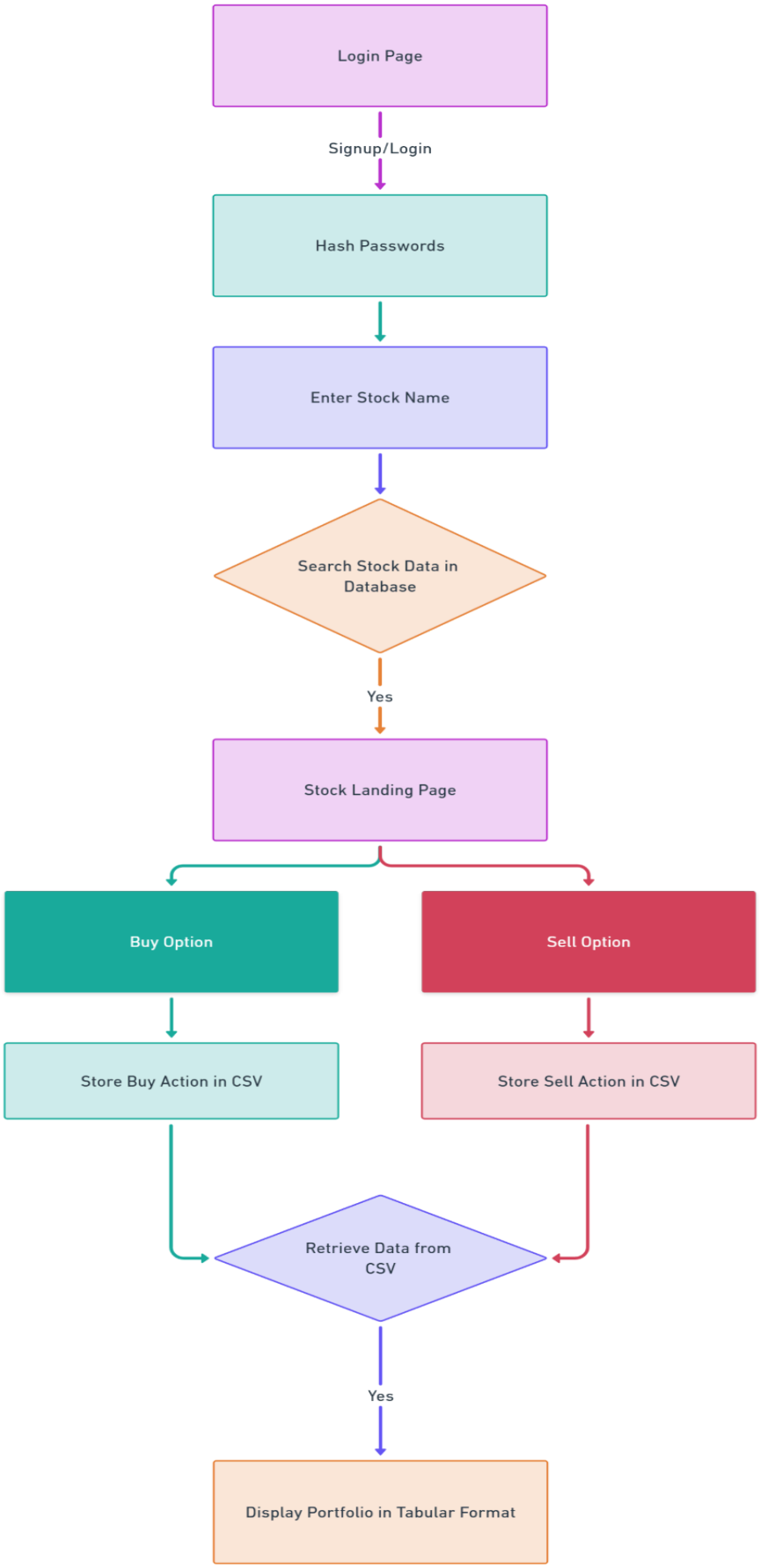
Hardware Specifications

Microprocessor (CPU)	: i5-12400F 2.5GHz 6 Cores
Memory (RAM)	: 16 GB DDR4 3200MHz
Architecture	: 64-bit
Storage	: 1 TB NVMe SSD
Display	: LG Ultragear 144Hz
Keyboard	: Logitech G213
Mouse	: Razer DeathAdder v2
GPU	: ZOTAC NVIDIA GeForce RTX 3050 8GB GDDR6
Printer	: Inkjet/Laser

Software Specifications

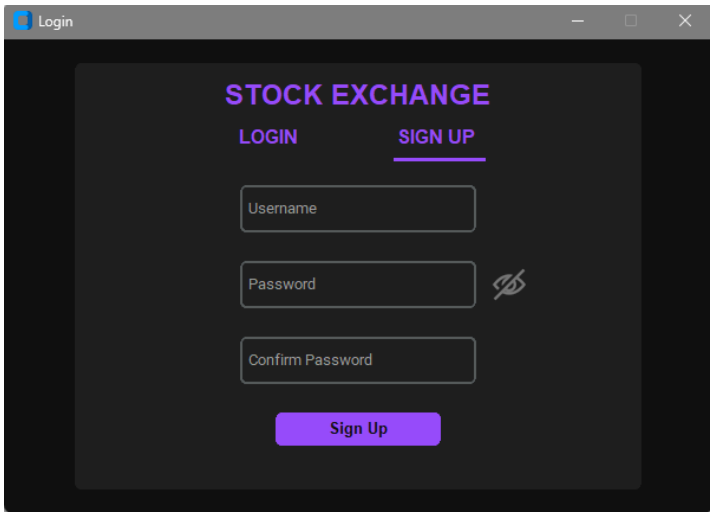
Operating System	: Windows 11 Enterprise
Front-End Design	: Python 3.12.1
Back-End	: Python 3.12.1, JSON & CSV
Documentation	: Microsoft Word 2016
Version Control	: Git

5. Context Diagram

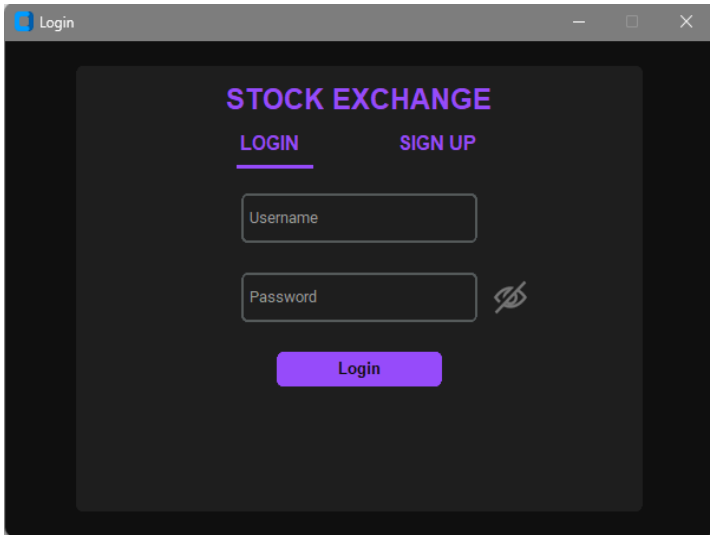


5. SignUp & Login Page

- **On running the Application, the Login Page will open.**



- **Now click on "SIGN UP" to create a new account.**
- **Enter your desired Username and Password.**
- **Click "Sign Up".**



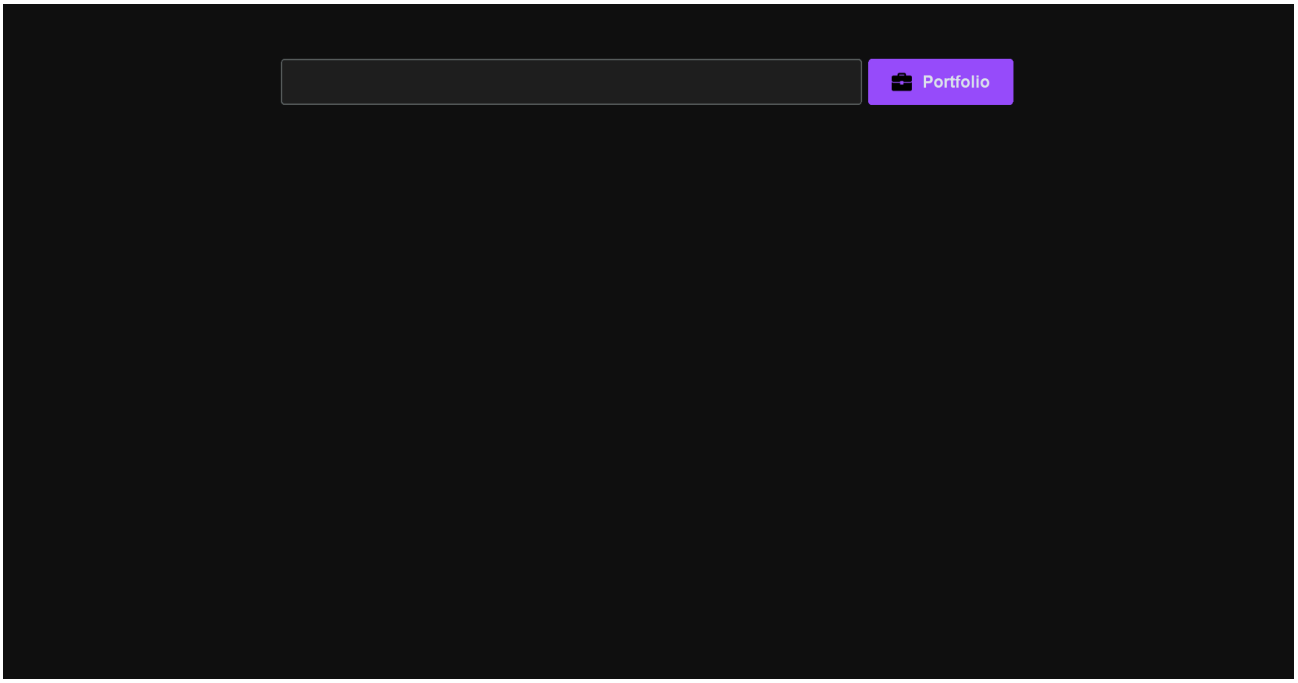
- **Enter the Registered Username.**
- **Enter the Password.**
- **Click "Login".**

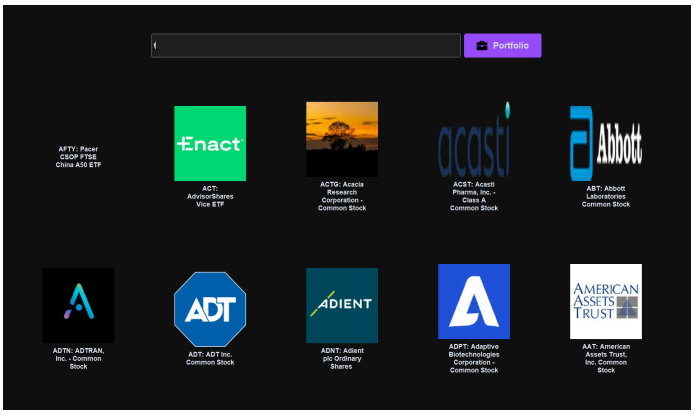
★ *On entering Incorrect Username/Password or leaving any Data Entry empty will raise a warning asking for the necessary details.*

★ *For enhanced **Security** the **Passwords** and **Usernames** have been **hashed** using **Military Grade-SHA-256**.*

6. Landing Page

- **After logging in the Home Page can be seen.**



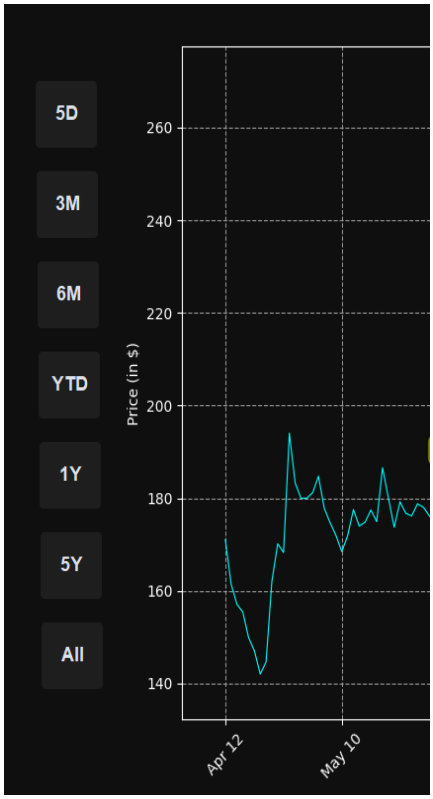


- **The Search Bar can be used to search for Stocks using Stock Names or Tickers / Stock Symbols.**

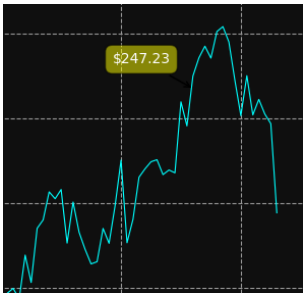
★ ***Entering invalid Stock Names / Tickers will not yield any results.***

7. Stock Page

- **Click on the Desired Stock to view the Stock Page.**
- **The stock's page will display the stock's details such as Opening Price, Closing Price, Business Summary etc.**



- **Hover near the Left End of the Price Chart to view the various Time Periods available.**
- **"All" displays the Price Data since the Stock's Listing.**
- **"YTD" displays the stock's Year to Date data of the Stock for the current Financial Year.**

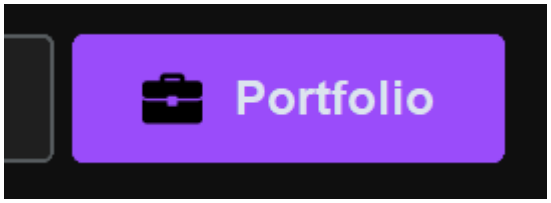


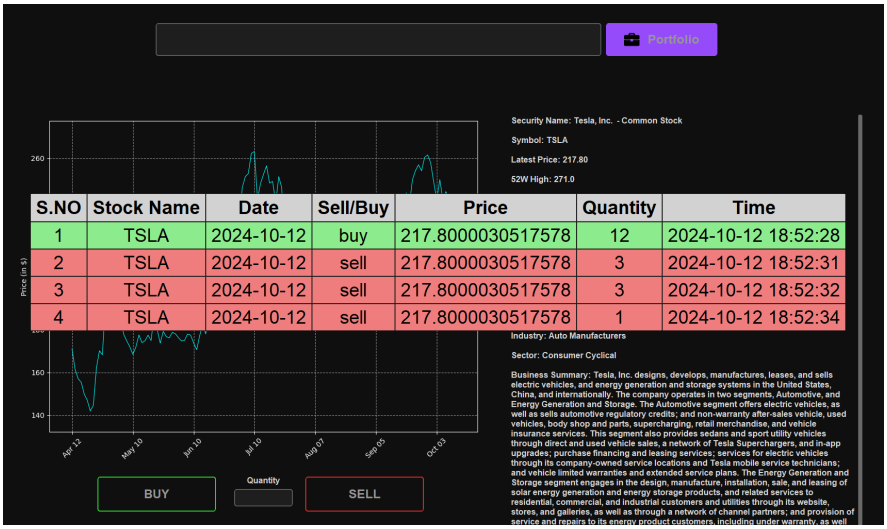
- **Hover over the Chart to see the exact Stock Prices at a given Time.**

- **To view the Website (if it exists), double-click the highlighted link.**

- **To Buy/Sell Stocks, enter the desired Quantity and Click the "BUY" or "SELL" Button.**

- **The change in Stock Holdings will be displayed in the Holdings Section which can be visited using the Portfolio Button.**





- **Suppose you bought 12 Tesla Stocks and Sold 3, 3, 1 of them respectively, the Holdings would look like the above example.**

  (If no Quantity is Entered, then it will Buy/Sell 1 Stock only).

- **The Holdings Screen can be closed by pressing the "Escape" Key.**

- **The Application can be closed in the same way by pressing "Escape".**

8. Source Code

a) Code - I :-

```
import os


try:
    import customtkinter as ctk
    from PIL import Image
    from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
    import matplotlib.pyplot as plt
    import webbrowser
    import validators
except:
    os.system("pip install customtkinter")
    os.system("pip install pillow")
    os.system("pip install matplotlib")
    os.system("pip install webbrowser")
    os.system("pip install validators")


    import customtkinter as ctk
    from PIL import Image
    from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
    import matplotlib.pyplot as plt
    import webbrowser
```

```
    import validators


from time import sleep
import json
import hashlib
import sys
import threading
import time
from mpl_finance import search_from_name, get_ticker_details,
make_graph
from buy_sell import record_transaction, create_portfolio


ctk.set_appearance_mode("dark")
ctk.set_default_color_theme("dark-blue")


data_loc = f"{os.path.dirname(os.path.realpath(__file__))}\\Data"
database_loc = f"{data_loc}\\Account Data Base.json"


win = ctk.CTk(fg_color="#121212")
win.title("Login")
win.geometry("0x0")


w = win.winfo_screenwidth()
h = win.winfo_screenheight()
```

```
pos = [15]*7
sys.setrecursionlimit(1500)

#Quit Application

def quit_application(event):

    if event.keysym == "Escape":
        win.destroy()

#Quit Portfolio

def quit_portfolio(event, frame, buttons):

    if event.keysym == "Escape":
        frame.destroy()
        win.unbind("<KeyPress>")
        for btn in buttons:
            btn.configure(state="normal")
        win.bind("<KeyPress>", quit_application)

#Center Window Method

def center(win, screen_resolution, animation_time):
```

```
    x1, y1 = 0, 0

    while x1 != screen_resolution[0] or y1 != screen_resolution[1]:

        if x1 != screen_resolution[0]: x1 +=
screen_resolution[0]/(animation_time*10)
        if y1 != screen_resolution[1]: y1 +=
screen_resolution[1]/(animation_time*10)

        win.geometry(f"{int(x1)}x{int(y1)}")

        x2 = w//2 - win.winfo_width()//2
        y2 = h//2 - win.winfo_height()//2

        win.geometry(f"+{x2}+{y2}")

        sleep(0.008)
        win.update()

#Login and Sign Up Functions

#Switch Between Login and Sign Up Windows
def switch_method(button):

    def make_selector(x, length):
```

```
        for i in range(1, length):

            text = "_"*i

            selector.configure(text=text)
            selector.place(x=x)

            sleep(0.04)

            win.update()

    if username.get() != "":
        username.delete(0, ctk.END)
    if password.get() != "":
        password.delete(0, ctk.END)

    try:
        pass_notification.destroy()
    except:
        pass

    check_var.set("")
    show_pass([password])

    if button == "login":
```

```
        login_page(False)
        make_selector(136, 6)

    else:

        signup_page()
        make_selector(270, 7)

#Show Password
def show_pass(inputs):

    show = check_var.get()

    if show == "*":
        check_var.set("")

show_password.configure(image=ctk.CTkImage(dark_image=Image.open(f"{dat
a_loc}\\Images\\Eye Show.png"), size=(30,30)))
    else:
        check_var.set("*")

show_password.configure(image=ctk.CTkImage(dark_image=Image.open(f"{dat
a_loc}\\Images\\Eye Hide.png"), size=(30,30)))

    for input in inputs:
```

```
        input.configure(show=check_var.get())


#Login Page
def login_page(first_time):


    global frame, selector, username, password, check_var,
show_password, login_button


    win.title("Login")


    try:
        confirm_password.destroy()
        signup_button.destroy()
    except:
        try:
            login_button.destroy()
        except:
            pass


    if first_time:


        center(win, (600, 400), 2)


        frame = ctk.CTkFrame(win, fg_color="#1f1f1f")
        frame.pack(pady=20, padx=60, fill="both", expand=True)
```

```
        label = ctk.CTkLabel(frame, text="STOCK EXCHANGE",
text_color="#9a4cfa", font=("Helvetica", 24, "bold"))
        label.pack(pady=12, padx=10)

    if first_time:

        selector = ctk.CTkLabel(frame, text="_____",
text_color="#9a4cfa", font=("Helvetica", 24, "bold"))
        selector.place(x=136, y=55)

    login_switch_method = ctk.CTkButton(frame, text="LOGIN",
text_color="#9a4cfa", font=("Helvetica", 16, "bold"),
fg_color="#1f1f1f", hover_color="#1f1f1f", cursor="hand2", height=0,
width=0, command=lambda: switch_method("login"))
    login_switch_method.place(x=136, y=50)

    sign_up_switch_method = ctk.CTkButton(frame, text="SIGN UP",
text_color="#9a4cfa", font=("Helvetica", 16, "bold"),
fg_color="#1f1f1f", hover_color="#1f1f1f", cursor="hand2", height=0,
width=0, command=lambda: switch_method("sign_up"))
    sign_up_switch_method.place(x=270, y=50)

    if first_time:

        username = ctk.CTkEntry(frame, placeholder_text="Username",
fg_color="#1f1f1f", show="", height=40, width=200)
```

```
        username.pack(pady=(50, 12), padx=10)


        check_var = ctk.StringVar()

        check_var.set("*")


        password = ctk.CTkEntry(frame, placeholder_text="Password",
fg_color="#1f1f1f", show="*", height=40, width=200)

        password.pack(pady=12, padx=10)


        show_password = ctk.CTkButton(frame, text="",
fg_color="#1f1f1f", hover_color="#191919",
image=ctk.CTkImage(dark_image=Image.open(f"{data_loc}\\Images\\Eye
Hide.png"), size=(30,30)), width=30, height=30, command=lambda:
show_pass([password]))

        show_password.place(x=345, y=168)


    else:


        show_password.configure(command=lambda: show_pass([password]))


    login_button = ctk.CTkButton(frame, text="Login",
text_color="#121212", fg_color="#9a4cfa", hover_color="#4937D2",
font=("Helvetica", 13, "bold"), cursor="hand2", command=login)

    login_button.pack(pady=12, padx=10)


    win.bind("<Return>", lambda event: login())
```

```
#Sign Up Page
def signup_page():

    global confirm_password, signup_button, return_key_bind

    win.title("Sign Up")

    try:
        login_button.destroy()
    except:
        try:
            confirm_password.destroy()
            signup_button.destroy()
        except:
            pass

    confirm_password = ctk.CTkEntry(frame, placeholder_text="Confirm
Password", fg_color="#1f1f1f", show="*", height=40, width=200)
    confirm_password.pack(pady=12, padx=10)

    show_password.configure(command=lambda: show_pass([password,
confirm_password]))

    signup_button = ctk.CTkButton(frame, text="Sign Up",
text_color="#121212", fg_color="#9a4cfa", hover_color="#4937D2",
font=("Helvetica", 13, "bold"), cursor="hand2", command=sign_up)
```

```
    signup_button.pack(pady=12, padx=10)

    return_key_bind = win.bind("<Return>", lambda event: sign_up())

#Login Button Function
def login():

    global user, pass_notification, account

    try:

        with open(database_loc, "r") as f:

            database = json.load(f)

    except:

        database = ""

    if (username.get() != "") and (password.get() != ""):

        user = hashlib.sha256(username.get().encode()).hexdigest()
        passw = hashlib.sha256(password.get().encode()).hexdigest()
```

```
        if (user in database) and (database[user] == passw): text = "Login Successful!"; color = "green"; x = 90
        else: text = "Username or Password is incorrect."; color = "red"; x = 92

        pass_notification = ctk.CTkLabel(frame, text=text, font=("Helvetica", 10, "bold"), text_color=color, height=0, width=300)
        pass_notification.place(x=x, y=212)

        frame.after(5000, pass_notification.destroy)

        if text == "Login Successful!":
            path = f"{data_loc}\\Accounts\\{user}"

            os.makedirs(path, exist_ok=True)

            account = path

            stock_collection()

    else:

        if username.get() == "": text = "You have a Username right?"; x = 92
        else: text = "I believe you forgot the Password"; x = 91
```

```
        pass_notification = ctk.CTkLabel(frame, text=text,
font=("Helvetica", 10, "bold"), text_color="red", height=0, width=300)

        pass_notification.place(x=x, y=212)


        frame.after(5000, pass_notification.destroy)


#Signup Button Function

def sign_up(username_taken = False):


    global pass_notification


    if (password.get() == confirm_password.get()) and (password.get()
!= "" and confirm_password.get() != "" and username.get() != "") and
(not username_taken):


        user = hashlib.sha256(username.get().encode()).hexdigest()

        passw = hashlib.sha256(password.get().encode()).hexdigest()


        if not os.path.exists(database_loc):


            with open(database_loc, 'w+') as f: pass


        with open(database_loc, "r") as f:


            try: database = json.load(f)

            except: database = ""
```

```
        if database == "":

            dict = {user: passw}

            with open(database_loc, "w+") as f: json.dump(dict, f,
indent=4)

            pass_notification = ctk.CTkLabel(frame, text="Registered
Successfully", font=("Helvetica", 10, "bold"), text_color="green",
height=0, width=300)
            pass_notification.place(x=89, y=276)

            frame.after(5000, pass_notification.destroy)

        elif user not in database:

            dict = database
            dict[user] = passw

            with open(database_loc, "w+") as f: json.dump(dict, f,
indent=4)

            pass_notification = ctk.CTkLabel(frame, text="Registered
Successfully", font=("Helvetica", 10, "bold"), text_color="green",
height=0, width=300)
```

```
            pass_notification.place(x=89, y=276)

            frame.after(5000, pass_notification.destroy)

        else:

            username_taken = True
            sign_up(username_taken)

        if not username_taken:
            os.makedirs(f"{data_loc}\\Accounts\\{user}", exist_ok=True)

    else:

        if username.get() == "": text = "Don't you need a Username?"; x
= 140
        elif (password.get() == "") and (confirm_password.get() == ""):
text = "Trust Me. You need a Password"; x = 140
        elif password.get() != confirm_password.get(): text = "The
Passwords do not Match"; x = 140
        else: text = "Username is already taken"; x = 140

        pass_notification = ctk.CTkLabel(frame, text=text,
font=("Helvetica", 10, "bold"), text_color="red", height=0, width=200)
        pass_notification.place(x=x, y=276)
```

```
        frame.after(5000, pass_notification.destroy)


#Display Portfolio
def display_portfolio(search):


    buttons = [widget for widget in search.winfo_children()]


    for btn in buttons:
        btn.configure(state="disabled")


    win.unbind("<KeyPress>")


    portfolio_frame = ctk.CTkFrame(win)
    portfolio_frame.place(relx=0.5, rely=0.5, anchor=ctk.CENTER)


    win.bind("<KeyPress>", lambda event,
portfolio_frame=portfolio_frame: quit_portfolio(event, portfolio_frame,
buttons))


    size = h/30


    create_portfolio(portfolio_frame, data_loc, user, size)


#Book Collection
def stock_collection():
```

```
    global saved_widgets

    frame.destroy()
    win.unbind("<Return>")

    center(win, (w, h), 2)
    win.attributes('-fullscreen', True)

    search = ctk.CTkFrame(win, bg_color="#1f1f1f", fg_color="#0f0f0f",
corner_radius=0)
    search.pack(fill="both", expand=True)

    search_term = ctk.StringVar()
    search_term.trace_add('write', lambda var, index, mode:
run_thread(search, search_term))

    search_bar = ctk.CTkEntry(search, fg_color="#1f1f1f",
textvariable=search_term, font=("Helvetica", h/67.5, "bold"))
    search_bar.place(relx=0.44, rely=0.1, relwidth=0.45,
relheight=0.06, anchor=ctk.CENTER)

    portfolio = ctk.CTkButton(search, text=" Portfolio",
fg_color="#9a4cfa", hover_color="#4937D2", font=("Helvetica", h/45,
"bold"), cursor="hand2",
image=ctk.CTkImage(dark_image=Image.open(f"{data_loc}\\Images\\Portfoli
o.png"), size=(h/36, h/36)), compound="left", command=lambda:
display_portfolio(search))
```

```
    portfolio.place(in_=search_bar, relx=1.01, rely=0, relwidth=0.25,
relheight=1)


    saved_widgets = [search_bar, portfolio]


#Recommendation Threads
def run_thread(search, search_term):


    saved_widgets[1].configure(state="disabled")


    thread = threading.Thread(target=lambda: update_search(search,
search_term))
    thread.start()


#Killing Old Recommendations
def kill_recommendation(recommendation, search_term, search_text):


    while True:
        time.sleep(0.1)
        if search_text != search_term.get():
            recommendation.destroy()
            return


#Update Recommendations
def update_search(search, search_term):
```

```
    search_text = search_term.get()

    time.sleep(0.5)

    if (search_term.get() != search_text) or (len(search_term.get()) ==
0):
        saved_widgets[1].configure(state="normal")
        return

    recommendations = search_from_name(search_text)

    list_of_recommendations = []

    #Load Recommendations
    def load_recommendation(stock):

        try:
            os.system(f'curl.exe
"https://logo.clearbit.com/{get_ticker_details(stock)["Website"]}"
--output {stock}.png')

            image = Image.open(f"{stock}.png")
            recommendation = ctk.CTkButton(search, fg_color="#0f0f0f",
hover_color="#1f1f1f", compound=ctk.TOP, text=f"{stock}:
{recommendations[stock]}", width=w/8.53 + 25, height=h/4.32 + 65,
```

```
font=("Helevetica", h/60, "bold"), image=ctk.CTkImage(dark_image=image,
size=(h/5.4, h/5.4)), command=lambda stock=stock: get_stock(search,
search_term, stock, recommendations[stock]))
        except:
            recommendation = ctk.CTkButton(search, fg_color="#0f0f0f",
hover_color="#1f1f1f", compound=ctk.TOP, text=f"{stock}:
{recommendations[stock]}", width=w/8.53 + 25, height=h/4.32 + 65,
font=("Helevetica", h/60, "bold"), command=lambda stock=stock:
get_stock(search, search_term, stock, recommendations[stock]))


        try:
            os.remove(f"{stock}.png")
        except:
            pass


        recommendation._text_label.configure(wraplength=150)
        thread = threading.Thread(target=lambda
search_text=search_text: kill_recommendation(recommendation,
search_term, search_text))
        thread.start()


        list_of_recommendations.append(recommendation)


    threads = []


    for stock in recommendations:
```

```
        thread = threading.Thread(target=lambda stock=stock:
load_recommendation(stock))
        threads.append(thread)

    for thread in threads:
        thread.start()
    for thread in threads:
        thread.join()

    win.unbind("<Motion>")
    win.unbind("<Button-1>")

    for widget in search.winfo_children():
        if (widget not in saved_widgets) and (widget not in
list_of_recommendations):
            widget.destroy()

    iteration = 0
    relative_y = 0.2

    #Display Recommendations
    try:
        for recommendation in list_of_recommendations:

            if iteration % 5 == 0:
```

```
                recommendation.place(in_=search, relx=0.03,
rely=relative_y, relwidth=0.16, relheight=0.35)

                relative_y = 0.6

            else:

                recommendation.place(in_=prev_recommendation, relx=1.2,
rely=0, relwidth=1, relheight=1)


            prev_recommendation = recommendation

            iteration += 1


        saved_widgets[1].configure(state="normal")


    except:

        pass


#Display Desired Stock Details

def get_stock(search, search_term, ticker, security_name,
period="6mo"):

    global buy, sell


    plt.close("all")


    search_term.set("")


    for widget in search.winfo_children():

        if widget not in saved_widgets:
```

```
            widget.destroy()

    graph = make_graph(ticker, period, "1d")

    bg = ctk.CTkLabel(search, text="", fg_color="#0f0f0f")
    bg.place(relx=0.00, rely=0.15, relwidth=0.6, relheight=0.8)

    ticker_details = get_ticker_details(ticker)

    canvas = FigureCanvasTkAgg(graph, master=search)
    canvas.get_tk_widget().place(in_=bg, relheight=1, relwidth=1)

    text = f"Security Name: {security_name}\n\n"

    for title in ticker_details:
        text += f"{title}: {ticker_details[title]}\n\n"

    text_widget = ctk.CTkTextbox(search, font=("Helvetica", h/67.5,
"bold"), fg_color="#0f0f0f", wrap=ctk.WORD)
    text_widget.insert(ctk.END, text)
    text_widget.configure(state=ctk.DISABLED)
    text_widget.place(in_=bg, relx=0.95, rely=0.1, relheight=1,
relwidth=0.6)

    text_widget.tag_config("link", underline=1, foreground="#3366CC")
```

```
    try:
        website_line = list(ticker_details.keys()).index("Website")*2 +
3
        text_widget.tag_add("link", f"{website_line}.9",
f"{website_line}.0 lineend")
    except:
        pass

    def link(event):
        try:
            sel_start, sel_end = text_widget.tag_ranges("sel")
            text_widget.tag_remove(ctk.SEL, "1.0", ctk.END)
            selected_text = text_widget.get(sel_start, sel_end)
            if validators.url(selected_text):
                webbrowser.open_new_tab(selected_text)
        except:
            pass

    win.bind("<Button-1>", link)

    buy = ctk.CTkButton(search, text="BUY", font=("Helvetica", h/45,
"bold"), border_color="#32CD32", border_width=2, fg_color="#0f0f0f",
hover_color="#32CD32", cursor="hand2", command=lambda:
record_transaction(data_loc, user, "buy", ticker, quantity.get()))
```

```
    sell = ctk.CTkButton(search, text="SELL", font=("Helvetica", h/45,
"bold"), border_color="#D32F2F", border_width=2, fg_color="#0f0f0f",
hover_color="#D32F2F", cursor="hand2", command=lambda:
record_transaction(data_loc, user, "sell", ticker, quantity.get()))


    buy.place(in_=bg, relx=0.26, rely=0.92, relheight=0.08,
relwidth=0.2)
    sell.place(in_=buy, relx=1.75, rely=0, relheight=1, relwidth=1)


    #Validation Command
    def vcmd(char):
        return char.isdigit()


    validation_command = win.register(vcmd)


    quantity = ctk.CTkEntry(search, font=("Helvetica", h/67.5, "bold"),
justify="center", fg_color="#1f1f1f", validate="key",
validatecommand=(validation_command, "%S"))
    quantity.place(in_=buy, relx=1.15, rely=0.35, relheight=0.5,
relwidth=0.5)


    quantity_text = ctk.CTkLabel(search, text="Quantity",
font=("Helvetica", h/67.5, "bold"), justify="center")
    quantity_text.place(in_=quantity, relx=0, rely=-0.75,
relheight=0.55, relwidth=1)


    #Animate Graph Buttons
```

```
    def animate(x):
        if pos[x] >= 0:
            pos[x] -= 1
            win.update()
            win.after(20, lambda: animate(x))

    #Check Hover State for Graph
    def check_hover(event):
        global pos

        if (win.winfo_pointerx() > bg.winfo_rootx()) and
(win.winfo_pointerx() < bg.winfo_rootx()+bg.winfo_width()/8) and
(win.winfo_pointery() > bg.winfo_rooty()) and (win.winfo_pointery() <
bg.winfo_rooty()+bg.winfo_height()):

            x = 0

            prev_widget = placeholder
            for widget in place_order:

                widget.place(in_=prev_widget, x=-pos[x], rely=1.35,
relheight=1, relwidth=1)
                win.update()

                animate(x)
                x+=1
```

```
                prev_widget = widget

        else:

            for widget in place_order:
                widget.place_forget()
                pos = [15]*7

    #Recursive Call to Update Graph
    def update_graph(period):
        get_stock(search, search_term, ticker, security_name, period)

    #Graph Buttons
    placeholder = ctk.CTkLabel(search, text="", fg_color="#0f0f0f")
    placeholder.place(in_=bg, relx=0.05, rely=0.062, relheight=0.07,
relwidth=0.055)

    _5d = ctk.CTkButton(search, text="5D", font=("Helvetica", h/60,
"bold"), fg_color="#1f1f1f", hover_color="#3f3f3f", corner_radius=5,
command=lambda: update_graph("5d"))
    _3m = ctk.CTkButton(search, text="3M", font=("Helvetica", h/60,
"bold"), fg_color="#1f1f1f", hover_color="#3f3f3f", corner_radius=5,
command=lambda: update_graph("3mo"))
    _6m = ctk.CTkButton(search, text="6M", font=("Helvetica", h/60,
"bold"), fg_color="#1f1f1f", hover_color="#3f3f3f", corner_radius=5,
command=lambda: update_graph("6mo"))
```

```
    _ytd = ctk.CTkButton(search, text="YTD", font=("Helvetica", h/60,
"bold"), fg_color="#1f1f1f", hover_color="#3f3f3f", corner_radius=5,
command=lambda: update_graph("ytd"))

    _1y = ctk.CTkButton(search, text="1Y", font=("Helvetica", h/60,
"bold"), fg_color="#1f1f1f", hover_color="#3f3f3f", corner_radius=5,
command=lambda: update_graph("1y"))

    _5y = ctk.CTkButton(search, text="5Y", font=("Helvetica", h/60,
"bold"), fg_color="#1f1f1f", hover_color="#3f3f3f", corner_radius=5,
command=lambda: update_graph("5y"))

    _all = ctk.CTkButton(search, text="All", font=("Helvetica", h/60,
"bold"), fg_color="#1f1f1f", hover_color="#3f3f3f", corner_radius=5,
command=lambda: update_graph("max"))


    place_order = [_5d, _3m, _6m, _ytd, _1y, _5y, _all]


    win.bind("<Motion>", check_hover)


def main():


    win.resizable(False, False)


    login_page(True)


    win.bind("<KeyPress>", quit_application)


    win.mainloop()
```

```
if __name__ == "__main__":

    main()
```

b) Code - II :-

```
import os
import matplotlib.pyplot as plt

try:
    import mplcursors
    import yfinance as yf
    import mplfinance as mpf
    import pandas as pd

except:
    os.system("pip install mplfinance")
    os.system("pip install mplcursors")
    os.system("pip install yfinance")
```

```
    os.system("pip install pandas")


    import mplcursors
    import yfinance as yf
    import mplfinance as mpf
    import pandas as pd

data_loc = os.path.dirname(os.path.realpath(__file__))

stocks = pd.read_csv(f"{data_loc}/Data/Tickers.csv")

def get_stock_data(ticker, period, interval):
    hist = yf.download(ticker, period=period, interval=interval)
    return hist

def on_move(event):
    global x, y

    ax = event.inaxes
    if ax is not None:
        # convert x y device coordinates to axes data coordinates
        x, y = ax.transData.inverted().transform([event.x, event.y])

def make_graph(ticker, period, interval):
```

```
    data = get_stock_data(ticker, period, interval)

    custom_style = mpf.make_mpf_style(base_mpf_style='nightclouds',
facecolor="#0f0f0f")

    fig, axlist = mpf.plot(data, type="line", style=custom_style,
ylabel='Price (in $)', volume=False, linecolor="cyan", returnfig=True)

    mplcursors.cursor(fig, hover=True).connect("add", lambda sel:
sel.annotation.set_text(f"${sel.target[1]:.2f}"))

    fig.canvas.mpl_connect('motion_notify_event', on_move)

    rect = fig.patch
    rect.set_facecolor('#0f0f0f')

    return fig

def search_from_name(search, ticker=False):

    if ticker: recommendation_number = 1
    else: recommendation_number = 10

    symbols = {}
```

```
    for ticker in stocks[stocks["Symbol"].str.contains(search,
case=False)]["Symbol"][:recommendation_number]:
        symbols[ticker] = stocks[stocks["Symbol"] == ticker]["Security
Name"].values[0]

    for ticker in stocks[stocks["Security Name"].str.contains(search,
case=False)]["Symbol"][:recommendation_number-len(symbols)]:
        symbols[ticker] = stocks[stocks["Symbol"] == ticker]["Security
Name"].values[0]

    return symbols

def get_ticker_details(ticker):

    stock = yf.Ticker(ticker)

    info = stock.info

    latest_price = stock.history(period='5d',
interval='1d').tail(1)['Close'].values[0]

    info["latest_price"] = f"{latest_price:.2f}"

    required_details = {"Symbol": "symbol", "Latest Price":
"latest_price", "52W High": "fiftyTwoWeekHigh", "52W Low":
"fiftyTwoWeekLow", "Total Revenue": "totalRevenue", "Market Cap":
"marketCap", "Volume": "volume", "Country": "country", "State":
```

```
"state", "Website": "website", "Industry": "industry", "Sector":
"sector", "Business Summary": "longBusinessSummary"}


    results = {}


    for k, v in required_details.items():
        try:
            results[k] = info[v]
        except:
            pass


    return results


if __name__ == "__main__":


    ticker = list(search_from_name("GBIL").keys())[0]


    make_graph(ticker, "ytd", "1d")
    plt.show()


    for i in ([f"{k}: {v}" for k, v in
get_ticker_details(ticker).items()]):
        print(i)
```

c) Code - III :-

```
import customtkinter as ctk

import csv

import os

from datetime import datetime

import yfinance as yf


file_name = f"Portfolio.csv"


# Columns for the CSV, including Quantity

columns = ["S.NO", "Stock Name", "Date", "Sell/Buy", "Price",
"Quantity", "Time"]


def get_latest_stock_price(ticker):

    stock = yf.Ticker(ticker)
```

```
    latest_price = stock.history(period='5d',
interval='1d').tail(1)['Close'].values[0]

    return latest_price


def record_transaction(data_loc, username, transaction_type, ticker,
quantity):

    if quantity.isdigit(): quantity = int(quantity)

    else: quantity = 1


    current_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

    current_date = datetime.now().strftime('%Y-%m-%d')

    current_price = get_latest_stock_price(ticker)

    serial_number = 1


    # Create a filename based on the username

    file_path = f"{data_loc}/Accounts/{username}/{file_name}"


    # Create CSV file with headers if it doesn't exist

    if not os.path.exists(file_path):

        with open(file_path, 'w', newline='') as file:

            writer = csv.writer(file)

            writer.writerow(columns)


    # Read the existing data to find the last serial number

    with open(file_path, 'r') as file:
```

```
        reader = csv.reader(file)
        rows = list(reader)
        if len(rows) > 1:
            serial_number = int(rows[-1][0]) + 1

    # Append the new transaction
    with open(file_path, 'a', newline='') as file:
        writer = csv.writer(file)
        writer.writerow([serial_number, ticker, current_date,
transaction_type, current_price, quantity, current_time])

def load_csv(csv_file):
    data = []
    with open(csv_file, "r") as file:
        reader = csv.DictReader(file)
        for row in reader:
            data.append(row)
    return data

def render_data(frame, data, size):
    # column heading
    for col_index, col_name in enumerate(columns):
        label = ctk.CTkLabel(frame, text=col_name, font=("Helvetica",
size, "bold"), padx=10, pady=5, bg_color="lightgray",
text_color="black")
```

```
        label.grid(row=0, column=col_index, sticky="nsew", padx=1,
pady=1)  # Simulate border with padx and pady


    # load data
    for row_index, row in enumerate(data):
        for col_index, col_name in enumerate(columns):
            value = row.get(col_name, "")
            tag = 'buy' if row["Sell/Buy"].lower() == 'buy' else 'sell'
            color = 'lightgreen' if tag == 'buy' else 'lightcoral'
            label = ctk.CTkLabel(frame, text=value, font=("Helvetica",
size), bg_color=color, text_color="black", padx=10, pady=5)
            label.grid(row=row_index+1, column=col_index,
sticky="nsew", padx=1, pady=1)  # Simulate border with padx and pady


    # edit table
    for i in range(len(columns)):
        frame.grid_columnconfigure(i, weight=1)
    for i in range(len(data) + 1):  # +1 for head row
        frame.grid_rowconfigure(i, weight=1)


def create_portfolio(frame, data_loc, username, size=24):
    file_path = f"{data_loc}/Accounts/{username}/{file_name}"
    data = load_csv(file_path)
    render_data(frame, data, size
```