VENKATESHWAR GLOBAL SCHOOL

ACADEMIC YEAR: 2024-2025

STOCK GAME

ROLL NO :
NAME : AARAV RAI AND PRATYAKSH KWATRA
CLASS : XII
SUBJECT : COMPUTER SCIENCE
SUB CODE : 083
PROJECT GUIDE:
MS. STUTI GUPTA
PGT (CS)

CERTIFICATE

This is to certify that Aarav Rai and Pratyaksh Kwatra have successfully completed the project work entitled 'The Stock Game' in the subject of Computer Science, as per the regulations of CBSE for the Practical Examination in Class XII to be held at VENKATESHWAR GLOBAL SCHOOL.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to everyone who contributed to the successful completion of this project.

First and foremost, we would like to thank our teacher, **Ms. Stuti Gupta**, for her continuous support, guidance, and valuable feedback throughout this project. Her expertise and encouragement were instrumental in helping us understand the complexities of the subject and in motivating us to strive for excellence.

We extend our deepest appreciation to our school principal, **Dr. Namita Singhal** for providing a conducive learning environment and the necessary resources that made this project possible. Her commitment to fostering academic growth has been a significant source of inspiration.

PROJECT ON STOCK PREDICTION

This project involves developing an interactive stock market guessing game. The game is designed to help users engage with the stock market by testing their knowledge and intuition about stock prices and market trends. It features two distinct games that challenge players to guess the identity of stocks based on historical price data and predict future price movements.

The project leverages a combination of Python libraries and APIs to fetch real-time stock data, visualize stock price trends, and manage user data through a MySQL database. The game is implemented with a user-friendly interface that allows players to create accounts, track their scores, and competeon a leaderboard.

PROJECT OBJECTIVES

1. Educational Engagement: The primary objective is to provide a platform for users to learn about stock market behaviour in a fun and interactive way.

2. Real-Time Data Integration: The project integrates real-time stock market data using the Yahoo FinanceAPI, ensuring that users interact with up-to-date and relevant information.

3. Database Management: The project includes the creation and management of a MySQL database to store user credentials, scores, and game results, allowing for persistent data tracking.

4. User Authentication: Implement a secure user authentication system that allows users to create accounts, log in, and manage their account settings.

PROJECT OBJECTIVES

5. Interactive Gameplay: Develop two unique games that challenge users to guess stock identities based on historical data and predict future stock movements, enhancing their understanding of market trends.

6. Leaderboard and Score Tracking: Maintain a leaderboard to foster competition among users, and track individual scores to monitor user progress and engagement over time.

By achieving these objectives, the project aims to provide an enjoyable learning experience that bridges the gap between theoretical knowledge and real-world stock market analysis

main.py

```python
import os
import subprocess
import sys

def check_requirements():
    if not os.path.isfile("requirements.txt"):
        print("Error: requirements.txt not found. Exiting program.")
        sys.exit(1)

    with open("requirements.txt") as f:
        packages = f.read().splitlines()

    for package in packages:
        try:
            __import__(package.split('==')[0])
        except ImportError:
            print(f"{package} not found. Installing...")
            subprocess.check_call([sys.executable, "-m", "pip", "install", package])

    print("All requirements are satisfied.")

check_requirements()

from mysql.connector import connect as mysql_connect, Error as mysql_Error
from yfinance import download as yf_download, Ticker as yf_Ticker
from nsepython import nse_eq_symbols
from matplotlib.pyplot import figure as plt_figure, plot as plt_plot, title as plt_title, xlabel as plt_xlabel, ylabel as plt_ylabel, show as plt_show
from random import sample, choice, randint, random
from datetime import datetime, timedelta
from hashlib import sha256
import keys
import inquirer3
```

Main.py

```
db_config = {
    'user': 'root',
    'password': keys.password,
    'host': 'localhost',
    'database': 'stock_game'
}

def get_db_connection():
    try:
        conn = mysql_connect(**db_config)
        return conn
    except mysql_Error as err:
        print(f"Error: {err}")
        return None

def setup_database():
    conn = mysql_connect(user='root', password=db_config['password'], host='localhost')
    if conn:
        cursor = conn.cursor()

        try:
            with open("database_integration.sql", "r") as file:
                sql_code_lines = [line.replace("\n", "") + ";" if line else None for line in file.read().split(";")]

                for statement in sql_code_lines:
                    if statement:
                        cursor.execute(statement)
                        conn.commit()
        except Exception as e:
            print("Error: Missing some files")
            print(e)
            quit()

        cursor.close()
        conn.close()

```

```
70 def login_menu():
71     global user
72     questions = [
73         inquirer3.List(
74             'auth_choice',
75             message="What would you like to proceed with?",
76             choices=['Create User', 'Login'],
77         ),
78     ]
79     choice = inquirer3.prompt(questions)["auth_choice"]
80     username = input("Enter a username: ")
81     password = input("Enter a password: ")
82     if choice == "Create User":
83         user = create_user(username, password)
84         if user:
85             main_menu()
86     elif choice == "Login":
87         user = login_user(username, password)
88         if user:
89             main_menu()
90         else:
91             print("Login failed.")
92     else:
93         print("Invalid choice.")
94 
95 def create_user(username, password):
96     hashed_password = sha256(password.encode()).hexdigest()
97     conn = get_db_connection()
98     cursor = conn.cursor()
99     try:
100         cursor.execute("INSERT INTO users (username, password) VALUES (%s,
    %s)", (username, hashed_password))
101         conn.commit()
102         cursor.execute("SELECT id, score FROM users WHERE username = %s",
    (username,))
103         user = cursor.fetchone()
104         print("User created successfully!")
105         return {'id': user[0], 'score': user[1]}
106     except mysql_Error as err:
107         print(f"Error: {err}")
108         return None
109     finally:
110         cursor.close()
111         conn.close()
```

Main.py

```
def login_user(username, password):
    hashed_password = sha256(password.encode()).hexdigest()
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute("SELECT id, score FROM users WHERE username = %s AND password = %s", (username, hashed_password))
    user = cursor.fetchone()
    cursor.close()
    conn.close()
    if user:
        print("Login successful!")
        return user
    else:
        print("Invalid credentials")
        login_menu()
        return None

def delete_user(user_id):
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute(f"DELETE FROM scores WHERE user_id = {user_id}")
    cursor.execute(f"DELETE FROM users WHERE id = {user_id}")
    conn.commit()
    cursor.close()
    conn.close()
    print("Account deleted successfully!")
    login_menu()

def update_username(user_id, new_username):
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("UPDATE users SET username = %s WHERE id = %s", (new_username, user_id))
    conn.commit()
    cursor.close()
    conn.close()
    print("Username changed successfully!")

```

Main.py

```python
def update_score(user_id, game_type, score):
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("UPDATE users SET score = score + %s WHERE id = %s",
(score, user_id))
    cursor.execute("INSERT INTO scores (user_id, game_type, score,
date_played) VALUES (%s, %s, %s, %s)",
                   (user_id, game_type, score, datetime.now()))
    conn.commit()
    cursor.close()
    conn.close()

def view_leaderboard():
    global user
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("SELECT username, score FROM users where score != 0 order
by score desc")
    result = cursor.fetchall()
    for players in result:
        print(f"Player: {players[0]}, Total Score: {players[1]}")
    conn.commit()
    cursor.close()
    conn.close()

def view_score():
    global user
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute(f"SELECT score FROM users WHERE id = {user['id']}")
    result = cursor.fetchone()
    total_score = result[0] if result[0] is not None else 0
    print("Total Score:", total_score)
    conn.commit()
    cursor.close()
    conn.close()
```

Main.py

```
global_stocks = [
    'AAPL', 'MSFT', 'GOOGL', 'AMZN', 'META',
    'TSLA', 'BRK-B', 'NVDA', 'JPM', 'JNJ',
    'V', 'WMT', 'PG', 'DIS', 'MA',
    'UNH', 'HD', 'VZ', 'PYPL', 'ADBE'
]
nse_small = [
    'RELIANCE.NS', 'TCS.NS', 'INFY.NS', 'HDFCBANK.NS', 'ICICIBANK.NS',
    'KOTAKBANK.NS', 'SBIN.NS', 'HDFC.NS', 'BHARTIARTL.NS', 'ITC.NS',
    'LT.NS', 'HCLTECH.NS', 'AXISBANK.NS', 'MARUTI.NS', 'ASIANPAINT.NS',
    'ULTRACEMCO.NS', 'M&M.NS', 'SUNPHARMA.NS', 'BAJFINANCE.NS',
]TITAN.NS'
nse_entirety = nse_eq_symbols()

def fetch_stock_data(ticker, start=None, end=None):
    if start and end:
        return yf_download(ticker, start=start, end=end, progress=False)
    return yf_download(ticker, period='max', progress=False)

def plot_stock_data(stock_data, title):
    plt_figure(figsize=(10, 5))
    plt_plot(stock_data['Close'])
    plt_title(title)
    plt_xlabel('Date')
    plt_ylabel('Close Price')
    plt_show(block=True)

def get_stock_info(ticker):
    stock_info = yf_Ticker(ticker).info
    name = stock_info.get('shortName', ticker)
    industry = stock_info.get('industry', 'Unknown')
    return name, industry
```

main.py

```python
def game1(user, stock_list):
    select_stocks = sample(stock_list, 4)
    select_stocks_nse = [stc + '.NS' for stc in select_stocks]
    if stock_list == nse_entirety:
        selected_stocks = select_stocks_nse
    else:
        selected_stocks = select_stocks
    correct_stock = choice(selected_stocks)
    correct_stock_full = get_stock_info(choice(selected_stocks))[0]

    end_date = datetime.now()
    start_date = end_date - timedelta(days=3*365)

    stock_data = fetch_stock_data(correct_stock, start=start_date,
end=end_date)

    _, industry = get_stock_info(correct_stock)

    plot_stock_data(stock_data, f"Guess the Stock (Industry: {industry})")

    import time
    time.sleep(2)
    questions = [
        inquirer3.List(
            'game_stock_choice',
            message="Guess a stock: ",
            choices=[get_stock_info(stock)[0] for stock in selected_stocks],
        ),
    ]
    guess = inquirer3.prompt(questions)["game_stock_choice"]

    if guess == correct_stock_full:
        print("Correct!")
        update_score(user['id'], 'game1', 10)
    else:
        print(f"Wrong! The correct answer was {correct_stock_full}.")
        update_score(user['id'], 'game1', 0)
```

```python
def game2(user, stock_list):
    select_stocks = sample(stock_list, 4)
    select_stocks_nse = [stc + '.NS' for stc in select_stocks]
    if stock_list == nse_entirety:
        selected_stocks = select_stocks_nse
    else:
        selected_stocks = select_stocks
    correct_stock = choice(selected_stocks)
    stock_data = fetch_stock_data(correct_stock)

    stock_name, industry = get_stock_info(correct_stock)

    start_date = stock_data.index[0]
    end_date = stock_data.index[-1]
    duration_years = 3

    def get_random_start_date(first_date, last_date, duration_years):
        max_start_date = last_date - timedelta(days=duration_years*365)

        try:
            random_start_date = first_date + timedelta(
                days=randint(0, (max_start_date - first_date).days)
            )
        except:
            return None

        return random_start_date

    random_start_date = get_random_start_date(start_date, end_date,
duration_years)

    if random_start_date is None:
        return

    random_end_date = random_start_date + timedelta(days=duration_years*365)

    three_year_data = stock_data[random_start_date:random_end_date]

    one_year_later_start = random_end_date + timedelta(days=1)
    one_year_later_end = one_year_later_start + timedelta(days=365)
    one_year_data = stock_data[one_year_later_start:one_year_later_end]
    average_price_one_year_later = one_year_data['Close'].mean().item()
```

```
296 if random() < 0.8:
297         display_title = f"Guess the Stock Movement (Stock: {stock_name},
    Industry: {industry})"
298     else:
299         bonus = True
300         print("Bonus Question! Guess without the stock name for extra
    points!")
301         display_title = f"Guess the Stock Movement (Industry: {industry})"
302 
303     final_price = three_year_data['Close'].iloc[-1].item()
304 
305     if average_price_one_year_later > final_price:
306         correct_answer = "up"
307     else:
308         correct_answer = "down"
309 
310     if average_price_one_year_later > final_price * 1.3:
311         correct_answer = "up a lot"
312     elif average_price_one_year_later < final_price * 0.7:
313         correct_answer = "down a lot"
314 
315     plot_stock_data(three_year_data, display_title)
316 
317     questions = [
318         inquirer3.List(
319             'choice',
320             message="Did the stock go up or down after the 3 year period?",
321             choices=['up a lot', 'up', 'down', 'down a lot'],
322         ),
323     ]
324     guess = inquirer3.prompt(questions)["choice"]
325 
326     if guess == correct_answer:
327         print("Precisely Correct!")
328         update_score(user['id'], 'game2', 15)
329         if 'bonus' in locals() and bonus == True:
330             print("Extra 5 points for answering bonus question!")
331             update_score(user['id'], 'game2', 5)
332     else:
333         if guess == (correct_answer + " a lot") or correct_answer == (guess +
    " a lot"):
334             print("Correct!")
335             update_score(user['id'], 'game2', 10)
336         else:
337             print(f"Wrong! The correct answer was {correct_answer}.")
338             update_score(user['id'], 'game2', 0)
```

```
336 def main():
337     setup_database()
338     login_menu()
339     while True:
340         main_menu()
341 
342 def main_menu():
343     questions = [
344         inquirer3.List(
345             'main_menu_choice',
346             message="What would you like to proceed with?",
347             choices=['Play', 'Account Settings', "View Leaderboard", "View
    Score", "Quit"],
348         ),
349     ]
350     choice = inquirer3.prompt(questions)["main_menu_choice"]
351 
352     if choice == "Play":
353         game_menu()
354     elif choice == "Account Settings":
355         account_menu()
356     elif choice == "View Leaderboard":
357         view_leaderboard()
358     elif choice == "View Score":
359         view_score()
360     elif choice == "Quit":
361         quit()
362     else:
363         print("Invalid Choice")
364         main_menu()
365 
366 def account_menu():
367     global user
368     questions = [
369         inquirer3.List(
370             'account_menu_choice',
371             message="What would you like to proceed with?",
372             choices=['Delete Account', 'Update Username', "Main Menu",
    "Quit"],
373         ),
374     ]
375     menu_choice = inquirer3.prompt(questions)["account_menu_choice"]
376     if menu_choice == "Delete Account":
377         delete_user(user['id'])
378     elif menu_choice == "Update Username":
379         new_username = input("Enter new username:")
380         update_username(user['id'], new_username)
381     elif menu_choice == "Main Menu":
382         return
383     elif menu_choice == "Quit":
384         quit()
385     else:
386         print("Invalid Choice")
387         return
```

Main.py

```python
def game_menu():
    global user

    questions = [
        inquirer3.List(
            'game_menu_choice',
            message="What would you like to proceed with?",
            choices=['Game 1', 'Game 2', "Main Menu", "Quit"],
        ),
    ]
    game_choice = inquirer3.prompt(questions)
["game_menu_choice"]
    if game_choice == "Game 1":
        while True:
            game1(user, global_stocks)
    elif game_choice == "Game 2":
        while True:
            game2(user, nse_entirety)
    elif game_choice == "Main Menu":
        return
    elif game_choice == "Quit":
        quit()
    else:
        print("Invalid Choice")
        return

if __name__ == "__main__":
    main()
```

database_integration.sql

```sql
CREATE DATABASE IF NOT EXISTS stock_game;

USE stock_game;

CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL,
    score INT DEFAULT 0
);

CREATE TABLE IF NOT EXISTS scores (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    game_type VARCHAR(10),
    score INT,
    date_played DATETIME,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

OUTPUT

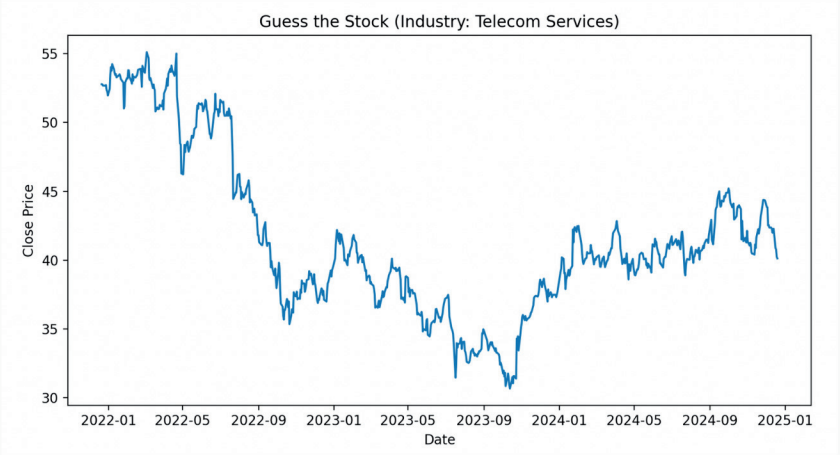
```
mysql-connector-python not found. Installing...
Defaulting to user installation because normal site-pa
Requirement already satisfied: mysql-connector-python
WARNING: You are using pip version 21.2.4; however, ve
You should consider upgrading via the '/Library/Develo
All requirements are satisfied.
[?] What would you like to proceed with?: Create User
 > Create User
   Login
```

```
Enter a username: test
Enter a password: 1234
User created successfully!
```

```
[?] What would you like to proceed with?: Login
   Create User
 > Login

Enter a username: test
Enter a password: 1234
Login successful!
```

```
[?] What would you like to proceed with?: Play
 > Play
   Account Settings
   View Leaderboard
   View Score
   Quit
```

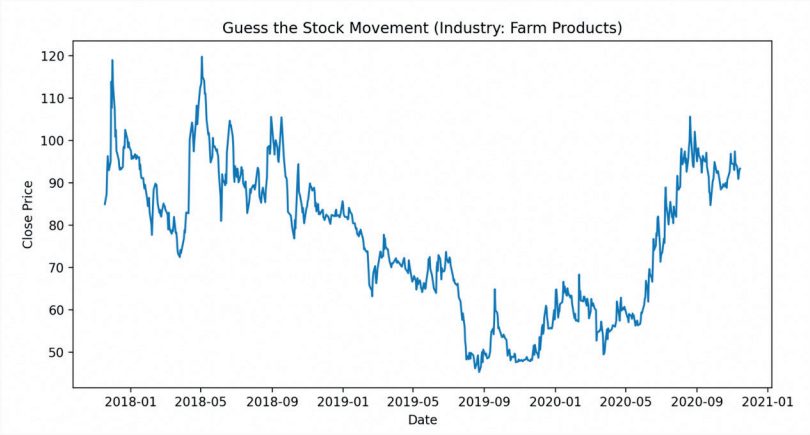
```
[?] What would you like to proceed with?: Game 1
 > Game 1
   Game 2
   Main Menu
   Quit
```




```
[?] Guess a stock: : Meta Platforms, Inc.
   Adobe Inc.
   Berkshire Hathaway Inc. New
 > Meta Platforms, Inc.
   Microsoft Corporation
```

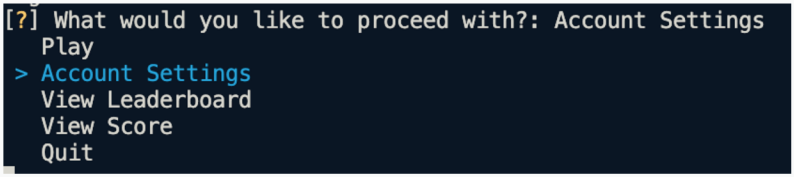
```
Wrong! The correct answer was Adobe Inc..
```

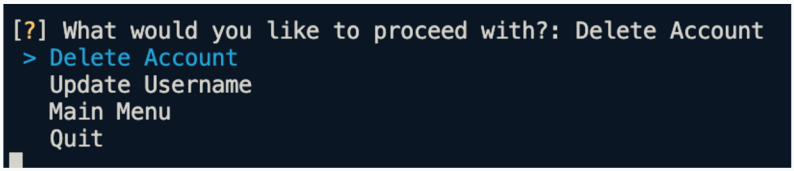
```
[?] What would you like to proceed with?: Game 2
   Game 1
 > Game 2
   Main Menu
   Quit
```

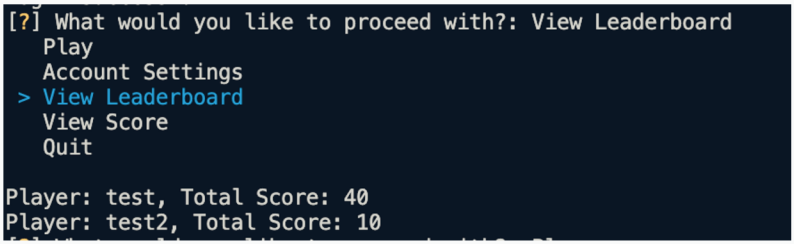


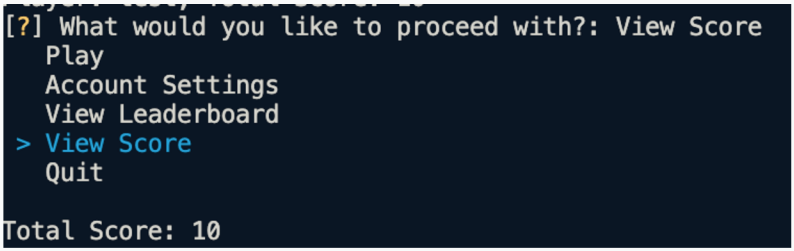

```
[?] Did the stock go up or down after the 3 year period?: up a lot
 > up a lot
   up
   down
   down a lot

Correct!
```

```
Bonus Question! Guess without the stock name for extra points!
[?] Did the stock go up or down after the 3 year period?: up a lot
 > up a lot
   up
   down
   down a lot
```


[?] What would you like to proceed with?: Account Settings
Play
> Account Settings
View Leaderboard
View Score
Quit


[?] What would you like to proceed with?: Delete Account
> Delete Account
Update Username
Main Menu
Quit


[?] What would you like to proceed with?: View Leaderboard
Play
Account Settings
> View Leaderboard
View Score
Quit
Player: test, Total Score: 40
Player: test2, Total Score: 10


[?] What would you like to proceed with?: View Score
Play
Account Settings
View Leaderboard
> View Score
Quit
Total Score: 10

```
mysql> SELECT * FROM users;
+----+----------+--------------+-------+
| id | username | password     | score |
+----+----------+--------------+-------+
|  1 | test     | 03ac674216f4 |    10 |
+----+----------+--------------+-------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM scores;
+----+---------+-----------+-------+---------------------+
| id | user_id | game_type | score | date_played         |
+----+---------+-----------+-------+---------------------+
|  1 |       1 | game1     |     0 | 2024-12-20 00:48:42 |
|  2 |       1 | game1     |     0 | 2024-12-20 00:48:55 |
|  3 |       1 | game1     |     0 | 2024-12-20 00:49:03 |
|  4 |       1 | game1     |     0 | 2024-12-20 00:49:13 |
|  5 |       1 | game2     |     0 | 2024-12-20 00:49:49 |
|  6 |       1 | game2     |     0 | 2024-12-20 00:49:58 |
|  7 |       1 | game2     |     0 | 2024-12-20 00:50:04 |
|  8 |       1 | game2     |     0 | 2024-12-20 00:50:13 |
|  9 |       1 | game2     |     0 | 2024-12-20 00:52:50 |
| 10 |       1 | game2     |     0 | 2024-12-20 00:52:54 |
| 11 |       1 | game1     |    10 | 2024-12-20 00:54:33 |
+----+---------+-----------+-------+---------------------+
11 rows in set (0.01 sec)
```


Thank
You!