

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 compete on 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 Finance API, 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.

SOURCE CODE



main.py

```
1 import os
2 import subprocess
3 import sys
4
5 def check_requirements():
6     if not os.path.isfile("requirements.txt"):
7         print("Error: requirements.txt not found. Exiting program.")
8         sys.exit(1)
9
10    with open("requirements.txt") as f:
11        packages = f.read().splitlines()
12
13    for package in packages:
14        try:
15            __import__(package.split('==')[0])
16        except ImportError:
17            print(f"{package} not found. Installing...")
18            subprocess.check_call([sys.executable, "-m", "pip", "install",
19                                  package])
20
21    print("All requirements are satisfied.")
22
23 check_requirements()
24
25 from mysql.connector import connect as mysql_connect, Error as mysql_Error
26 from yfinance import download as yf_download, Ticker as yf_Ticker
27 from nsepython import nse_eq_symbols
28 from matplotlib.pyplot import figure as plt_figure, plot as plt_plot, title as
29     plt_title, xlabel as plt_xlabel, ylabel as plt_ylabel, show as plt_show
30 from random import sample, choice, randint, random
31 from datetime import datetime, timedelta
32 from hashlib import sha256
33 import keys
34 import inquirer
```



Main.py

```
34 db_config = {  
35     'user': 'root',  
36     'password': keys.password,  
37     'host': 'localhost',  
38     'database': 'stock_game'  
39 }  
40  
41 def get_db_connection():  
42     try:  
43         conn = mysql_connect(**db_config)  
44         return conn  
45     except mysql_Error as err:  
46         print(f"Error: {err}")  
47         return None  
48  
49 def setup_database():  
50     conn = mysql_connect(user='root', password=db_config['password'],  
    host='localhost')  
51     if conn:  
52         cursor = conn.cursor()  
53  
54     try:  
55         with open("database_integration.sql", "r") as file:  
56             sql_code_lines = [line.replace("\n", "") + ";" if line else  
None for line in file.read().split(";")]  
57  
58         for statement in sql_code_lines:  
59             if statement:  
60                 cursor.execute(statement)  
61                 conn.commit()  
62     except Exception as e:  
63         print("Error: Missing some files")  
64         print(e)  
65         quit()  
66  
67     cursor.close()  
68     conn.close()  
69
```



```
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,
101 %s)", (username, hashed_password))
102         conn.commit()
103         cursor.execute("SELECT id, score FROM users WHERE username = %s",
104 (username,))
105         user = cursor.fetchone()
106         print("User created successfully!")
107         return {'id': user[0], 'score': user[1]}
108     except mysql_Error as err:
109         print(f"Error: {err}")
110         return None
111     finally:
112         cursor.close()
113         conn.close()
```



```
113 def login_user(username, password):
114     hashed_password = sha256(password.encode()).hexdigest()
115     conn = get_db_connection()
116     cursor = conn.cursor(dictionary=True)
117     cursor.execute("SELECT id, score FROM users WHERE username = %s AND
password = %s", (username, hashed_password))
118     user = cursor.fetchone()
119     cursor.close()
120     conn.close()
121     if user:
122         print("Login successful!")
123         return user
124     else:
125         print("Invalid credentials")
126         login_menu()
127         return None
128
129 def delete_user(user_id):
130     conn = get_db_connection()
131     cursor = conn.cursor()
132     cursor.execute(f"DELETE FROM scores WHERE user_id = {user_id}")
133     cursor.execute(f"DELETE FROM users WHERE id = {user_id}")
134     conn.commit()
135     cursor.close()
136     conn.close()
137     print("Account deleted successfully!")
138     login_menu()
139
140 def update_username(user_id, new_username):
141     conn = get_db_connection()
142     cursor = conn.cursor()
143     cursor.execute("UPDATE users SET username = %s WHERE id = %s",
(new_username, user_id))
144     conn.commit()
145     cursor.close()
146     conn.close()
147     print("Username changed successfully!")
148
```



```
149 def update_score(user_id, game_type, score):
150     conn = get_db_connection()
151     cursor = conn.cursor()
152     cursor.execute("UPDATE users SET score = score + %s WHERE id = %s",
153                   (score, user_id))
153     cursor.execute("INSERT INTO scores (user_id, game_type, score,
154 date_played) VALUES (%s, %s, %s, %s)",
155                   (user_id, game_type, score, datetime.now()))
156     conn.commit()
157     cursor.close()
158     conn.close()
159
160 def view_leaderboard():
161     global user
162     conn = get_db_connection()
163     cursor = conn.cursor()
164     cursor.execute("SELECT username, score FROM users where score != 0 order
165 by score desc")
166     result = cursor.fetchall()
167     for players in result:
168         print(f"Player: {players[0]}, Total Score: {players[1]}")
169     conn.commit()
170     cursor.close()
171     conn.close()
172
173 def view_score():
174     global user
175     conn = get_db_connection()
176     cursor = conn.cursor()
177     cursor.execute(f"SELECT score FROM users WHERE id = {user['id']}")
```



```
184 global_stocks = [
185     'AAPL', 'MSFT', 'GOOGL', 'AMZN', 'META',
186     'TSLA', 'BRK-B', 'NVDA', 'JPM', 'JNJ',
187     'V', 'WMT', 'PG', 'DIS', 'MA',
188     'UNH', 'HD', 'VZ', 'PYPL', 'ADBE'
189 ]
190 nse_small = [
191     'RELIANCE.NS', 'TCS.NS', 'INFY.NS', 'HDFCBANK.NS', 'ICICIBANK.NS',
192     'KOTAKBANK.NS', 'SBIN.NS', 'HDFC.NS', 'BHARTIARTL.NS', 'ITC.NS',
193     'LT.NS', 'HCLTECH.NS', 'AXISBANK.NS', 'MARUTI.NS', 'ASIANPAINT.NS',
194     'ULTRACEMCO.NS', 'M&M.NS', 'SUNPHARMA.NS', 'BAJFINANCE.NS',
195 ]TITAN.NS'
196 nse_entirety = nse_eq_symbols()
197
198 def fetch_stock_data(ticker, start=None, end=None):
199     if start and end:
200         return yf_download(ticker, start=start, end=end, progress=False)
201     return yf_download(ticker, period='max', progress=False)
202
203 def plot_stock_data(stock_data, title):
204     plt.figure(figsize=(10, 5))
205     plt.plot(stock_data['Close'])
206     plt.title(title)
207     plt.xlabel('Date')
208     plt.ylabel('Close Price')
209     plt.show(block=True)
210
211 def get_stock_info(ticker):
212     stock_info = yf.Ticker(ticker).info
213     name = stock_info.get('shortName', ticker)
214     industry = stock_info.get('industry', 'Unknown')
215     return name, industry
```



```
217 def game1(user, stock_list):
218     select_stocks = sample(stock_list, 4)
219     select_stocks_nse = [stc + '.NS' for stc in select_stocks]
220     if stock_list == nse_entirety:
221         selected_stocks = select_stocks_nse
222     else:
223         selected_stocks = select_stocks
224     correct_stock = choice(selected_stocks)
225     correct_stock_full = get_stock_info(choice(selected_stocks))[0]
226
227     end_date = datetime.now()
228     start_date = end_date - timedelta(days=3*365)
229
230     stock_data = fetch_stock_data(correct_stock, start=start_date,
231                                    end=end_date)
232
233     _, industry = get_stock_info(correct_stock)
234
235     plot_stock_data(stock_data, f"Guess the Stock (Industry: {industry})")
236
237     import time
238     time.sleep(2)
239     questions = [
240         inquirer3.List(
241             'game_stock_choice',
242             message="Guess a stock: ",
243             choices=[get_stock_info(stock)[0] for stock in selected_stocks],
244         ),
245     ]
246     guess = inquirer3.prompt(questions)["game_stock_choice"]
247
248     if guess == correct_stock_full:
249         print("Correct!")
250         update_score(user['id'], 'game1', 10)
251     else:
252         print(f"Wrong! The correct answer was {correct_stock_full}.")
253         update_score(user['id'], 'game1', 0)
```



```
254 def game2(user, stock_list):
255     select_stocks = sample(stock_list, 4)
256     select_stocks_nse = [stc + '.NS' for stc in select_stocks]
257     if stock_list == nse_entirety:
258         selected_stocks = select_stocks_nse
259     else:
260         selected_stocks = select_stocks
261     correct_stock = choice(selected_stocks)
262     stock_data = fetch_stock_data(correct_stock)
263
264     stock_name, industry = get_stock_info(correct_stock)
265
266     start_date = stock_data.index[0]
267     end_date = stock_data.index[-1]
268     duration_years = 3
269
270     def get_random_start_date(first_date, last_date, duration_years):
271         max_start_date = last_date - timedelta(days=duration_years*365)
272
273         try:
274             random_start_date = first_date + timedelta(
275                 days=randint(0, (max_start_date - first_date).days))
276         )
277         except:
278             return None
279
280         return random_start_date
281
282     random_start_date = get_random_start_date(start_date, end_date,
283                                              duration_years)
284
285     if random_start_date is None:
286         return
287
288     random_end_date = random_start_date + timedelta(days=duration_years*365)
289
290     three_year_data = stock_data[random_start_date:random_end_date]
291
292     one_year_later_start = random_end_date + timedelta(days=1)
293     one_year_later_end = one_year_later_start + timedelta(days=365)
294     one_year_data = stock_data[one_year_later_start:one_year_later_end]
295     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},
298     Industry: {industry})"
299     else:
300         bonus = True
300     print("Bonus Question! Guess without the stock name for extra
301     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 +
334             " 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
```



```
389 def game_menu( ):
390     global user
391
392     questions = [
393         inquirer3.List(
394             'game_menu_choice',
395             message="What would you like to proceed with?",
396             choices=['Game 1', 'Game 2', "Main Menu", "Quit"],
397         ),
398     ]
399     game_choice = inquirer3.prompt(questions)
400 [ "game_menu_choice"]
401     if game_choice == "Game 1":
402         while True:
403             game1(user, global_stocks)
404     elif game_choice == "Game 2":
405         while True:
406             game2(user, nse_entirety)
407     elif game_choice == "Main Menu":
408         return
409     elif game_choice == "Quit":
410         quit()
411     else:
412         print("Invalid Choice")
413         return
414
415 if __name__ == "__main__":
416     main()
```



database_integration.sql

```
1 CREATE DATABASE IF NOT EXISTS stock_game;
2
3 USE stock_game;
4
5 CREATE TABLE IF NOT EXISTS users (
6     id INT AUTO_INCREMENT PRIMARY KEY,
7     username VARCHAR(50) UNIQUE NOT NULL,
8     password VARCHAR(100) NOT NULL,
9     score INT DEFAULT 0
10 );
11
12 CREATE TABLE IF NOT EXISTS scores (
13     id INT AUTO_INCREMENT PRIMARY KEY,
14     user_id INT,
15     game_type VARCHAR(10),
16     score INT,
17     date_played DATETIME,
18     FOREIGN KEY (user_id) REFERENCES users(id)
19 );
```

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
```

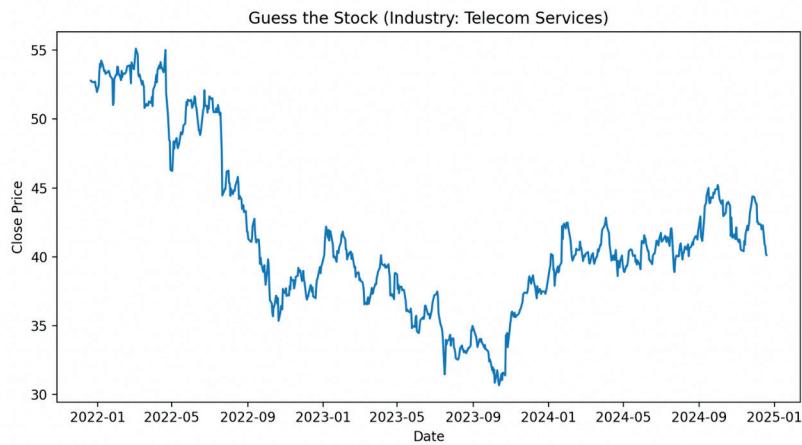
```
Enter a username: test
Enter a password: 1234
User created successfully!
```

```
[?] 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
```

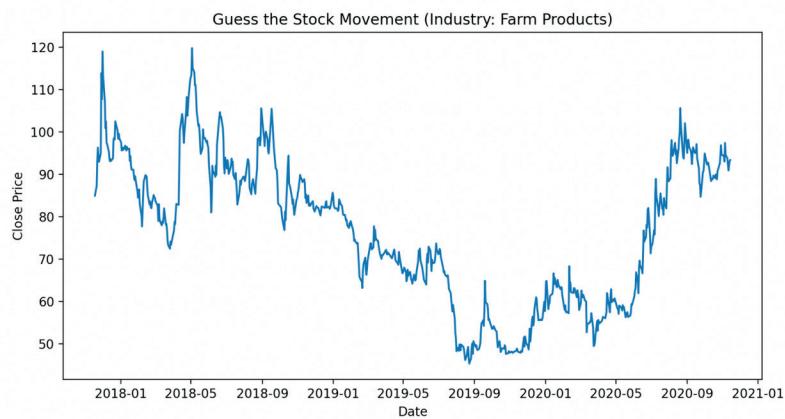
```
[?] What would you like to proceed with?: Game 1  
> Game 1  
Game 2  
Main Menu  
Quit
```



```
[?] 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!

