ระบบจัดการข้อมูลตลาดซื้อ-ขายนักเตะของทีม Big 6 ใน พรีเมียร์ลีก
(Big 6 Transfer Market Management System)

นายวุฒิเมธี กัณหารี **รหัสนักศึกษา 6706022610080**
นายกฤษดิณ เศรษฐโชดึก **รหัสนักศึกษา 6706022610110**
นายพสุธร ปรุงเกียรติ **รหัสนักศึกษา 6706022610187**

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตร์บัณฑิต
สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ
คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ
ปีการศึกษา 2568

คำนำ

การจัดทำโครงงาน “ระบบจัดการข้อมูลตลาดซื้อขายนักแตะ” นี้เป็นส่วนหนึ่งของ วิชาการเขียนโปรแกรมคอมพิวเตอร์ (Computer Programming) รหัสวิชา 060223115 ภาคเรียนที่ 1 ปี

การศึกษา 2568 หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศคณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือเพื่อให้นักศึกษาได้นำความรู้ที่เรียนมาทั้งหมดมาประยุกต์ใช้ในการพัฒนาโปรแกรมที่ สามารถทำงานได้จริงโดยเน้นการออกแบบและเขียนโปรแกรมในภาษา Python ซึ่งเป็นภาษาที่ใช้ใน การศึกษาเป็นหลักของในวิชานี้ โดยโครงงานนี้จะช่วย การคิดวิเคราะห์และแก้ปัญหาทางเทคนิค เพื่อ เตรียมความพร้อมในการประกอบอาชีพด้านวิศวกรรมสารสนเทศและเครือข่ายในอนาคต หากมี ข้อผิดพลาดประการใด คณะผู้จัดทำต้องขออภัยไว้ ณ ที่นี้ด้วย

สารบัญ

บทที่ 1
บทนำ

1.1. วัตถุประสงค์ของโครงงาน

1.1.1 เพื่อพัฒนาระบบที่สามารถจัดการกับรายชื่อนักแตะในหกทีมใหญ่ของพรีเมียร์ลีกได้อย่างมีประสิทธิภาพ

1.1.2 เพื่อฝึกฝนทักษะการเขียนโปรแกรมด้วยภาษาคอมพิวเตอร์ (Python)

1.1.3 เพื่อเรียนรู้การจัดการข้อมูลและไฟล์

1.1.4 เพื่อเรียนรู้การทำงานร่วมกันเป็นทีม

1.2. ขอบเขตของโครงงาน

1.2.1 ระบบจัดการข้อมูลหนังสือจะมีฟังก์ชันพื้นฐาน ฟังก์ชัน 8 เช่น 1. เพิ่มรายชื่อนักเตะ 2. อัพเดตข้อมูลนักเตะ 3. ลบมูลนักเตะ 4. ดูข้อมูลนักเตะทั้งหมด 5. บันทึกการซื้อขายนักเตะ 6. ดูข้อมูลการซื้อขายนักเตะ 7. การแสดงรายการซื้อขายนักเตะ 8. ออกจากโปรแกรม

1.2.2 ระบบจัดการข้อมูลซื้อขายนักเตะ ประกอบด้วย 11 ฟิลด์ ได้แก่ 1. Tranfer ID 2. Player ID 3. Player Name

4. From_Team 5. To_Team 6. Age 7. Price_Buy (Feel(M)) 8. Position 9. Status 10. Active 11. Date

1.2.3 ระบบจัดการเก็บข้อมูลการซื้อขายนักเตะไว้ในไฟล์

text file ชื่อ data.txt ซึ่งมี ชื่อนักเตะ ตำแหน่ง จำนวนที่

ราคาที่ซื้อล่าสุด ราคาที่ซื้อมา มาจากทีมอะไร และ ไปที่ทีม

ไหน

1.2.4 ระบบจัดการซื้อขายนักเตะในหกทีมชั้นนำของพรีเมียร์หลีกเพื่อให้ผู้ใช้

สามารถดูการซื้อขายได้ง่ายและขอมูลที่สำคัญครบท่วนในครั้งเดียว

ประโยชน์ที่ได้รับ

1.2.1 พัฒนาระบบที่สามารถจัดการรายการซื้อขายนักเตะได้อย่างมีประสิทธิภาพ

1.2.2 พัฒนาทักษะการเขียนโปรแกรมคอมพิวเตอร์

1.2.3 เรียนรู้การจัดการข้อมูลและไฟล์

1.2.4 เรียนรู้การทำงานร่วมกันเป็นทีม

1.3. เครื่องมือที่คาดว่าจะต้องใช้

1.3.1 ภาษา Python

1.3.2 โปรแกรม Visual Studio Code

1.3.3 Microsoft Office

บทที่ 2
ระบบการจัดการซื้อขายนักเตะ

2.1. แฟ้มข้อมูลรายชื่อนักเตะ

แฟ้มข้อมูลนักเตะประกอบด้วย

ฟิลด์	ชนิด	ขนาด(bytes)	ตัวอย่าง
player_ID	int	5	3002
player_Name	str	40	Luis Diaz
position	str	5	LWF
age_Buy	int	5	25
price_Buy	float	10	50.0
current_Team_id	int	5	3(LIV)
Active	bool	1	False

ตารางที่ 2-1 แฟ้มข้อมูลรายชื่อนักเตะ

2.1.1. Player ID รหัสประจำตัวนักเตะ (player_ID)

Player ID รหัสประจำตัวนักเตะ ใช้สำหรับระบุและแยกแยะนักเตะแต่ละคนในระบบอย่างเฉพาะเจาะจง ไม่ให้ซ้ำกัน ใช้ในการอ้างอิงข้อมูลของนักเตะ เช่น ชื่อ ตำแหน่ง สโมสร สัญชาติ และข้อมูลการโอนย้าย เพื่อให้สามารถเชื่อมโยงกับตารางอื่น ๆ เช่น ตารางการซื้อขาย (Transfer) หรือสโมสร (Club) ได้สะดวกและแม่นยำ

2.1.2. Player Name ชื่อนักเตะ (player_Name)

Player Name ชื่อของนักเตะในรูปแบบข้อความ ใช้เพื่อแสดงผลและค้นหาในระบบ กำหนดความยาวไว้สูงสุด 40 ตัวอักษร เพื่อให้ครอบคลุมชื่อเต็มของนักเตะจากหลายประเทศ

2.1.3. Position ตำแหน่งในสนามของนักเตะ (position)

Position ตำแหน่งของนักเตะในสนาม เช่น CF (Center Forward), AMF (Attacking Midfielder), RWF (Right Wing Forward), CB (Center Back) เป็นต้น ใช้ในการจัดหมวดหมู่และคัดกรองข้อมูลนักเตะ

2.1.4. Age Buy อายุที่ซื้อนักเตะ (age_Buy)

Age Buy อายุของนักเตะในขณะที่ทำการซื้อหรือโอนเข้าทีม ใช้เพื่อวิเคราะห์มูลค่าการซื้อขายและแนวโน้มของนักเตะในตลาด

2.1.5. Price Buy ราคาที่ซื้อ (price_Buy)

Price Buy ราคาที่ทีมซื้อนักเตะเข้ามา (หน่วยเป็นล้านยูโรหรือปอนด์ตามระบบที่กำหนด) เป็นค่าทศนิยมเพื่อรองรับมูลค่าการซื้อขาย เช่น 50.0 หมายถึง 50 ล้านยูโร

2.1.6. Current Team ID รหัสทีมปัจจุบัน (current_Team_id)

Current Team ID รหัสทีมปัจจุบันที่นักเตะสังกัด เช่น 3 = Liverpool ใช้เชื่อมโยงกับไฟล์ทีม (team.dat) เพื่อแสดงชื่อทีมโดยอัตโนมัติ

2.1.7. Active สถานะของนักเตะ (active)

Active สถานะของนักเตะในระบบ — ถ้า True หมายถึงยังอยู่ในระบบหรือยังเล่นอยู่ในทีมปัจจุบัน ถ้า False หมายถึงถูกลบหรือย้ายออก (ใช้สำหรับ “Soft Delete” ไม่ลบนักเตะจริงออกจากไฟล์)

2.2. แฟ้มข้อมูลทีม Big 6

แฟ้มข้อมูลทีม Big 6 ประกอบด้วย

ฟิลด์	ชนิด	ขนาด(bytes)	ตัวอย่าง
Team_ID	int	5	3
Team_Name	Str	20	Liverpool
Team_Code	Str	5	LIV
Big6	Bool	1	1

ตารางที่ 2-2 แฟ้มข้อมูลทีม Big 6

2.2.1 Team ID รหัสประจำทีม (team_ID)

Team ID รหัสประจำทีม ใช้เป็นตัวเลขแทนแต่ละสโมสรในระบบ (1–6 สำหรับทีม Big 6 และ 99 สำหรับทีมอื่น ๆ) เช่น 3 หมายถึง Liverpool ใช้เป็น Primary Key เพื่อเชื่อมโยงกับนักเตะ (player.dat) และการย้ายทีม (transfer.dat)

2.2.2 Team Name ชื่อทีม (team_Name)

Team Name ชื่อเต็มของสโมสร กำหนดความยาวสูงสุด 20 ตัวอักษร ใช้แสดงผลในรายงานและเมนูต่าง ๆ

2.2.3 Team Code รหัสย่อของทีม (team_Code)

Team Code รหัสย่อของทีม ใช้แทนชื่อสโมสรในการแสดงผล เช่น บนหน้าจอ ตาราง หรือรายงาน เช่น ARS, CHE, LIV เป็นต้น

2.2.4 Big 6 ทีม Big 6 ในพรีเมียร์ลีก (big6)

Big 6 ตัวบ่งชี้ว่าทีมนี้อยู่ในกลุ่ม Big 6 หรือไม่ (True = ใช่, False = ไม่ใช่) ใช้สำหรับจัดหมวดหมู่และสร้างรายงานแยกเฉพาะทีมใหญ่

2.3. แฟ้มข้อมูลการซื้อ-ขายนักเตะ

แฟ้มข้อมูลการซื้อ-ขายนักเตะ

ฟิลด์	ชนิด	ขนาด(bytes)	ตัวอย่าง
transfer_ID	int	5	7001
transfer_Player_id	int	5	3001
from_Team_id	int	5	3
to_Team_id	Int	5	99
age_Sell	int	5	28
price_Sell_m	float	5	75.0
date_yyyymmdd	int	10	20250901
status	bool	1	True
active	bool	1	True

ตารางที่ 2-3 แฟ้มข้อมูลการซื้อ-ขายนักเตะ

2.3.1 transfer_ID

tranfer_ID คือรหัสการซื้อขายนักเตะ ใช้สำหรับระบุรายการการโอนย้ายนักเตะแต่ละครั้งอย่างเฉพาะเจาะจง เพื่อไม่ให้ซ้ำกันในระบบ ใช้ในการอ้างอิงข้อมูลการซื้อขายนักเตะในฐานข้อมูล เพื่อเชื่อมโยงกับข้อมูลอื่น ๆ เช่น ชื่อนักเตะ ต้นสังกัดเดิม สโมสรใหม่ ค่าตัว วันโอนย้าย เป็นต้น

2.3.2 transfer_Player_ID รหัสประจำตัวนักเตะ (transfer_Player_ID)

transfer_Player_ID รหัสประจำตัวนักเตะ ใช้สำหรับระบุและแยกแยะนักเตะแต่ละคนในระบบอย่างเฉพาะเจาะจง ไม่ให้ซ้ำกัน ใช้ในการอ้างอิงข้อมูลของนักเตะ เช่น ชื่อ ตำแหน่ง สโมสร สัญชาติ และข้อมูลการโอนย้าย เพื่อให้สามารถเชื่อมโยงกับตารางอื่น ๆ เช่น ตารางการซื้อขาย (Transfer) หรือสโมสร (Club) ได้สะดวกและแม่นยำ

2.3.3 From Team ID ซื้อจากทีมอะไร (from_Team_id)

From_Team คือ ซื้อจากทีมอะไร เป็นฟิลด์ที่ใช้ระบุชื่อสโมสรต้นสังกัดเดิมของนักเตะก่อนที่จะถูกโอนย้ายหรือซื้อขายไปยังทีมใหม่

2.3.4 To Team ID ขายให้ทีมอะไร (to_Team_id)

To Team ID คือ ขายให้ทีมอะไร เป็นฟิลด์ที่ใช้ระบุชื่อสโมสรปลายทางหรือทีมใหม่ที่นักเตะถูกโอนย้ายไปหลังจากการซื้อขาย ใช้เพื่อระบุปลายทางของนักเตะในการโอนย้ายแต่ละครั้ง ช่วยใน

การติดตามประวัติการซื้อขายนักเตะและใช้เชื่อมโยงข้อมูลกับตารางสโมสร (Club) หรือข้อมูลการโอนย้าย (Transfer) เพื่อการวิเคราะห์และออกรายงานได้อย่างครบถ้วน

2.3.5 Age Sell อายุที่ย้ายทีม (age_Sell)

อายุของนักเตะในขณะที่ถูกขายหรือย้ายทีม ใช้ในการวิเคราะห์ช่วงอายุที่เหมาะสมกับการซื้อขาย

2.3.6 Price Sell ราคาที่ขาย (price_Sell_m)

ราคาขายหรือค่าตัวของนักเตะ (หน่วยเป็นล้าน) เป็นตัวเลขทศนิยมเพื่อรองรับมูลค่าที่ไม่เป็นจำนวนเต็ม เช่น 75.0 หมายถึง 75 ล้านยูโร

2.3.7 Date วันที่ที่ทำการซื้อ-ขาย (date_yyyymmdd)

วันที่ทำการย้ายทีมในรูปแบบ YYYYMMDD เช่น 20250118 หมายถึงวันที่ 18 มกราคม 2025 ใช้เพื่อเรียงลำดับดีลตามเวลา

2.3.8 Status สถานะของดีล (status)

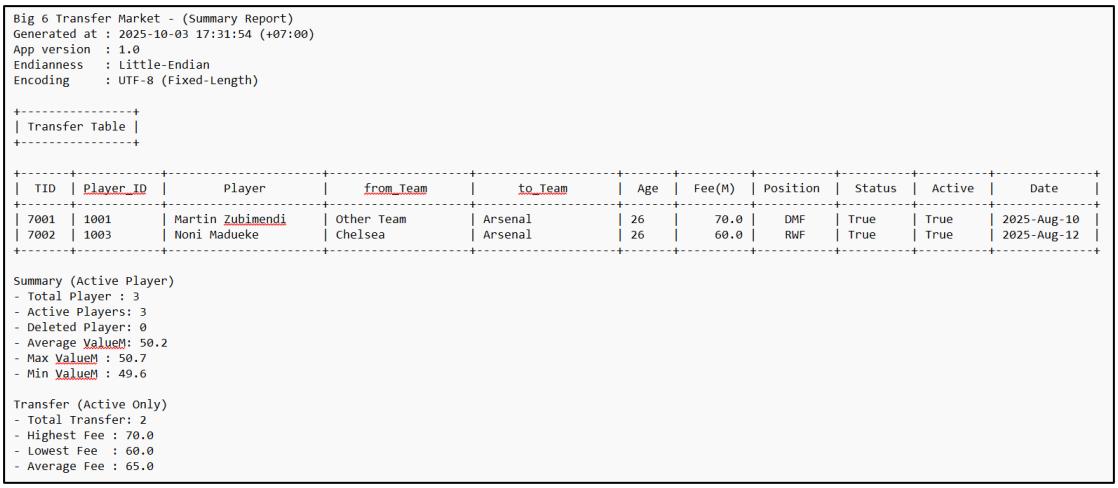
สถานะของดีลในเชิง “ผลการย้ายทีม” ถ้า True หมายถึง ดีลสำเร็จ, ถ้า False หมายถึง ดีลถูกยกเลิก

2.3.9 Active สถานะของนักเตะ (active)

สถานะของเรคอร์ดในระบบ ถ้า True หมายถึง ยังอยู่ในระบบ, ถ้า False หมายถึง ถูกลบออกจากระบบ (Soft Delete)

2.4. ไฟล์ report.txt

ไฟล์ report.txt ในระบบจัดการข้อมูลตลาดซื้อขายนักเตะของทีม Big 6 ใน พรีเมียร์ลีก ประกอบด้วย



```
Big 6 Transfer Market - (Summary Report)
Generated at : 2025-10-03 17:31:54 (+07:00)
App version  : 1.0
Endianness   : Little-Endian
Encoding     : UTF-8 (Fixed-Length)

+----------------+
| Transfer Table |
+----------------+

+------+-----------+--------------------+-------------+-------------+-----+----------+----------+--------+--------+-------------+
|  TID | Player_ID |       Player       |  from_Team  |   to_Team   | Age |  Fee(M)  | Position | Status | Active |     Date    |
+------+-----------+--------------------+-------------+-------------+-----+----------+----------+--------+--------+-------------+
| 7001 | 1001      | Martin Zubimendi   | Other Team  | Arsenal     | 26  |     70.0 |    DMF   | True   | True   | 2025-Aug-10 |
| 7002 | 1003      | Noni Madueke       | Chelsea     | Arsenal     | 26  |     60.0 |    RWF   | True   | True   | 2025-Aug-12 |
+------+-----------+--------------------+-------------+-------------+-----+----------+----------+--------+--------+-------------+

Summary (Active Player)
- Total Player : 3
- Active Players: 3
- Deleted Player: 0
- Average ValueM: 50.2
- Max ValueM : 50.7
- Min ValueM : 49.6

Transfer (Active Only)
- Total Transfer: 2
- Highest Fee : 70.0
- Lowest Fee  : 60.0
- Average Fee : 65.0
```

ภาพที่ 2-1 ไฟล์ report.txt

2.4.1 Report_Title

ชื่อรายงานหลัก แสดงหัวข้อของระบบและประเภทของรายงาน ใช้บ่งบอกว่าเป็นการสรุปข้อมูลการซื้อขายนักเตะในทีม Big 6

2.4.2 Generated_at

วันและเวลาที่โปรแกรมสร้างรายงาน ใช้ตรวจสอบความใหม่ของข้อมูล

2.4.3 App_version

เวอร์ชันของโปรแกรมที่สร้างรายงาน ใช้สำหรับติดตามการอัปเดตระบบ

2.4.4 Endianness

รูปแบบการเก็บข้อมูลในหน่วยความจำ (byte ล่างมาก่อน) เพื่อให้โปรแกรมอ่านข้อมูลตรงกัน

2.4.5 Encoding

การเข้ารหัสข้อความแบบ UTF-8 ความยาวคงที่ เพื่อให้ไฟล์ไบนารีอ่านค่าได้อย่างถูกต้อง

2.4.6 Transfer_Table

ตารางสรุปรายการย้ายทีมจากไฟล์ transfer.dat โดยเชื่อมข้อมูลกับ player.dat และ team.dat เพื่อแสดงรายละเอียดของแต่ละดีล

2.4.7 Summary_Active_Player

ส่วนสรุปข้อมูลของนักเตะที่ยัง Active ในระบบ เช่น จำนวน ค่าเฉลี่ย มูลค่าสูงสุด–ต่ำสุด

2.4.8 Summary_Transfer_Active

ส่วนสรุปข้อมูลของดีลย้ายทีมที่ยัง Active อยู่ เช่น จำนวนดีล ค่าตัวสูงสุด ต่ำสุด และเฉลี่ย

บทที่ 3

การใช้งานระบบจัดการข้อมูลตลาดซื้อ-ขายนักเตะของทีม Big 6 ใน พรีเมียร์ลีก

โปรแกรมจัดการซื้อขายนักเตะ Big 6 พรีเมียร์ลีก คือโปรแกรมที่ช่วยให้การจัดเก็บและบริหารข้อมูลการซื้อ–ขายนักเตะของสโมสรในกลุ่ม Big 6 พรีเมียร์ลีกอังกฤษ เป็นไปอย่างสะดวก รวดเร็ว และเป็นระบบมากขึ้น โดยโปรแกรมจะทำหน้าที่เก็บข้อมูลทีม นักเตะ และดีลการย้ายทีม พร้อมทั้งสามารถสรุปรายงานผลไปเก็บไว้ในไฟล์ข้อความ (Text File) เพื่อใช้ตรวจสอบภายหลังได้

สำหรับผู้ใช้งานโปรแกรม

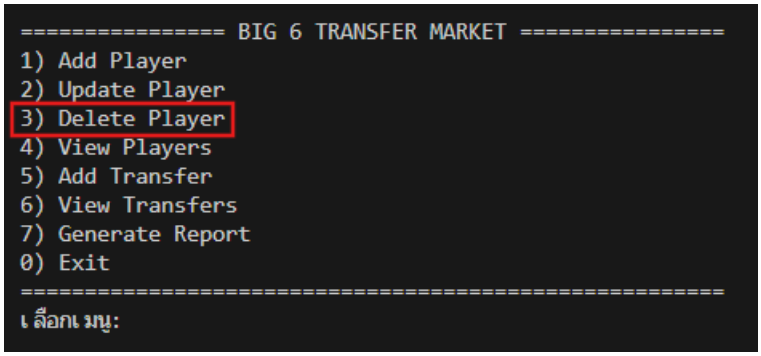
3.1. การใช้งานโปรแกรมระบบจัดการข้อมูลตลาดซื้อ-ขาย

3.1.1 รายละเอียดเมนูในฟังก์ชัน main_menu():

กด 1: เพิ่มข้อมูลนักเตะใหม่ (เรียกใช้ add_player())

กด 2: แก้ไขข้อมูลนักเตะ (เรียกใช้ update_player())

กด 3: ลบนักเตะ (Soft Delete โดยเปลี่ยนค่า active=False)



กด 4: แสดงรายชื่อนักเตะทั้งหมด (read_players())

```
================ BIG 6 TRANSFER MARKET ================
1) Add Player
2) Update Player
3) Delete Player
4) View Players
5) Add Transfer
6) View Transfers
7) Generate Report
0) Exit
=======================================================
เลือกเมนู:
```

กด 5: เพิ่มรายการซื้อขายนักเตะ (add_transfer())

```
================ BIG 6 TRANSFER MARKET ================
1) Add Player
2) Update Player
3) Delete Player
4) View Players
5) Add Transfer
6) View Transfers
7) Generate Report
0) Exit
=======================================================
เลือกเมนู:
```

กด 6: แสดงข้อมูลการซื้อขายทั้งหมด (read_transfers())

```
================ BIG 6 TRANSFER MARKET ================
1) Add Player
2) Update Player
3) Delete Player
4) View Players
5) Add Transfer
6) View Transfers
7) Generate Report
0) Exit
=======================================================
เลือกเมนู:
```

กด 7: สร้างรายงานสรุป (เรียกใช้ generate_report())

กด 0: ออกจากโปรแกรม

ฟังก์ชันนี้ใช้ **input()** เพื่อรับค่าจากผู้ใช้ ตรวจสอบความถูกต้องของตัวเลือก และเรียกฟังก์ชันที่เกี่ยวข้องตามหมายเลขที่เลือก

4.2 ฟังก์ชัน add_player()

ใช้สำหรับเพิ่มข้อมูลนักเตะใหม่ลงในไฟล์ players.dat โดยบันทึกข้อมูลในรูปแบบ fixed-length ผ่านไลบรารี struct เช่น

- Player_ID
- Name
- Position
- Age
- Price (ค่าตัว/ราคาซื้อ)
- Team_ID
- Active (สถานะการใช้งาน)

ช่วยให้สามารถเก็บข้อมูลนักเตะได้อย่างมีระบบและป้องกันการซ้ำของรหัสนักเตะ

4.3 ฟังก์ชัน update_player()

ใช้สำหรับแก้ไขข้อมูลนักเตะตาม **Player_ID** ที่ระบุ เช่น ชื่อ ตำแหน่ง อายุ ค่าตัว หรือทีมสังกัด โดยจะเปิดไฟล์ players.dat และเขียนทับเฉพาะระเบียนของนักเตะที่ต้องการแก้ไข

4.4 ฟังก์ชัน add_transfer()

ใช้บันทึกข้อมูลการโอนย้ายหรือซื้อขายนักเตะลงในไฟล์ transfers.dat เช่น

- Transfer_ID
- Player_ID
- From_Team
- To_Team
- Age
- Fee (ราคาค่าตัว)
- Date (วันที่ซื้อขาย)
- Status / Active

พร้อมอัปเดตทีมปัจจุบันของนักเตะในไฟล์ players.dat ให้ตรงกับทีมปลายทาง (To_Team) โดยอัตโนมัติ

4.5 ฟังก์ชัน read_players() และ read_transfers()

ใช้สำหรับอ่านข้อมูลจากไฟล์ไบนารี players.dat และ transfers.dat เพื่อนำมาแสดงผลหรือใช้ในการประมวลผลภายในระบบ เช่น การค้นหา การสร้างรายงาน และการตรวจสอบข้อมูล

4.6 ฟังก์ชัน generate_report()

ทำหน้าที่สร้างรายงานสรุป (summary.txt) ซึ่งประกอบด้วย

- ตารางข้อมูลการซื้อขายนักเตะทั้งหมด
- สถิติของนักเตะ (เช่น จำนวนที่ Active, ค่าเฉลี่ย, ค่าต่ำสุด–สูงสุดของราคานักเตะ)
- ข้อมูลการซื้อขายที่ยัง Active อยู่

รายงานจะอยู่ในรูปแบบข้อความตาราง (fixed-width table) อ่านง่าย เหมาะสำหรับตรวจสอบและวิเคราะห์ข้อมูลภายหลัง

4.7 โค้ดของฟังก์ชั่น Run()

```python
# ================================
# Menus
# ================================
def main_menu():
    while True:
        print("\n================ BIG 6 TRANSFER MARKET ================")
        print("1) Add Player")
        print("2) Update Player")
        print("3) Delete Player")
        print("4) View Players")
        print("5) Add Transfer")
        print("6) View Transfers")
        print("7) Generate Report")
        print("0) Exit")
        print("=======================================================")

        choice = input("เลือกเมนู: ").strip()
```

แสดงหัวข้อของโปรแกรม

เมื่อโปรแกรมเริ่มทำงาน ฟังก์ชัน run() (หรือในโค้ดคือ main_menu()) จะเริ่มต้นด้วยการแสดงหัวข้อของโปรแกรม เพื่อแจ้งให้ผู้ใช้ทราบว่าโปรแกรมได้เริ่มต้นทำงานแล้ว โดยใช้คำสั่ง print() เพื่อแสดงข้อความต้อนรับ

จากนั้นโปรแกรมจะแสดงชื่อโปรแกรมและเมนูหลักของระบบ เช่น

1. Add Player
2. Update Player
3. Delete Player
4. View Players
5. Add Transfer
6. View Transfers
7. Generate Report
8. Exit

ข้อความเหล่านี้ถูกใช้เพื่อให้ผู้ใช้เข้าใจหน้าที่ของแต่ละตัวเลือกอย่างชัดเจน ก่อนจะเริ่มรับคำสั่งจากผู้ใช้ผ่านฟังก์ชัน input() เพื่อดำเนินการในส่วนถัดไปของโปรแกรม โดยการจัดรูปแบบด้วยเส้นคั่นและการเว้นบรรทัดช่วยให้การแสดงผลมีความสวยงามและเป็นระเบียบ เพิ่มความเข้าใจและความสะดวกในการใช้งานของผู้ใช้

4.8 แสดงหัวข้อของฟังก์ชัน main()

```
def main_menu():
    while True:
        print("\n================ BIG 6 TRANSFER MARKET ================")
        print("1) Add Player")
        print("2) Update Player")
        print("3) Delete Player")
        print("4) View Players")
        print("5) Add Transfer")
        print("6) View Transfers")
        print("7) Generate Report")
        print("0) Exit")
        print("=======================================================")
```

ใช้คำสั่ง print() เพื่อ **แสดงชื่อหัวข้อของโปรแกรม** ว่า

“BIG 6 TRANSFER MARKET”

แสดงรายการเมนูทั้งหมดให้ผู้ใช้เลือก เช่น เพิ่มผู้เล่น (Add Player), ดูข้อมูล, ลบ, รายงาน ฯลฯ

ใช้ while True: เพื่อวนลูปเมนูตลอดเวลา จนกว่าผู้ใช้จะเลือก “0) Exit”

แสดงเมนูให้ผู้ใช้เลือกฟังก์ชัน run()

ตรวจสอบการเลือกของผู้ใช้

เมื่อผู้ใช้เลือกหมายเลขจากเมนู โปรแกรมจะตรวจสอบหมายเลขที่ผู้ใช้เลือกและเรียกใช้ฟังก์ชันที่สอดคล้องกับเมนูนั้น ๆ โดยใช้ **โครงสร้างควบคุมแบบ if-elif-else** (ในภาษา Python รุ่นใหม่สามารถใช้ match choice: ได้เช่นกัน) เพื่อทำการตรวจสอบค่าที่ผู้ใช้ป้อนเข้ามาและเรียกฟังก์ชันที่เกี่ยวข้องกับแต่ละเมนู

เคสที่ถูกเรียกใช้

1. **เพิ่มข้อมูลผู้เล่น (Add Player)**

   ⟶ เรียกฟังก์ชัน add_player() เพื่อเพิ่มข้อมูลนักเตะใหม่ลงในไฟล์ players.dat

2. **แก้ไขข้อมูลผู้เล่น (Update Player)**

   ⟶ เรียกฟังก์ชัน update_player() เพื่อแก้ไขข้อมูลของผู้เล่นที่มีอยู่

3. **ลบผู้เล่น (Delete Player)**

   ⟶ เรียกฟังก์ชัน update_player() พร้อมตั้งค่าสถานะ active=False เพื่อทำการลบผู้เล่นแบบ *Soft Delete*

4. **ดูข้อมูลผู้เล่น (View Players)**

   ⟶ เรียกฟังก์ชัน read_players() เพื่อแสดงข้อมูลผู้เล่นทั้งหมดจากไฟล์

5. **เพิ่มข้อมูลการย้ายทีม (Add Transfer)**

   ⟶ เรียกฟังก์ชัน add_transfer() เพื่อเพิ่มข้อมูลการย้ายทีมของนักเตะ

6. **ดูข้อมูลการย้ายทีม (View Transfers)**

   ⟶ เรียกฟังก์ชัน read_transfers() เพื่อแสดงข้อมูลการย้ายทีมทั้งหมด

7. **สร้างรายงานสรุป (Generate Report)**

   ⟶ เรียกฟังก์ชัน generate_report() เพื่อสร้างรายงานสรุปในไฟล์ summary.txt

8. **ตัวเลือกอื่น ๆ (ถ้าป้อนหมายเลขไม่ถูกต้อง)**

   ⟶ โปรแกรมจะแสดงข้อความแจ้งเตือนว่า “❌ เมนูไม่ถูกต้อง ลองใหม่อีกครั้ง” และวนกลับไปที่เมนูหลักอีกครั้ง

9. **ออกจากโปรแกรม (Exit)**

   ⟶ ถ้าผู้ใช้ป้อนหมายเลข 0 โปรแกรมจะแสดงข้อความ

```
if choice == "1":
    try:
        pid = int(input("Player ID: ").strip())
        ensure_unique_player_id(pid)
        name = input("Name: ").strip()
        pos  = require_position(input("Position (FW/MF/DF/GK/...): ").strip())
        age  = int(input("Age: ").strip())
        price= float(input("Price Buy (ล้าน): ").strip())
        team = require_team_id(int(input("Team ID (1-6 หรือ 99=OTH): ").strip()))
        add_player(pid, name, pos, age, price, team, active=True)
        print("✓ เพิ่ม Player สำเร็จ")
    except Exception as e:
        print(f"❌ {e}")

elif choice == "2":
    try:
        pid = int(input("Player ID ที่จะแก้ไข: ").strip())
        cur = find_player(pid)
        if not cur:
            print("❌ ไม่พบ Player ID นี้");  continue
        new_name = input(f"Name ใหม่ (Enter = {cur['name']}): ").strip()
        new_pos  = input(f"Position ใหม่ (Enter = {cur['pos']}): ").strip()
        new_age  = input(f"Age ใหม่ (Enter = {cur['age']}): ").strip()
        new_price= input(f"Price ใหม่ (Enter = {fmt_money(cur['price'])}): ").strip()
        new_team = input(f"Team ID ใหม่ (Enter = {cur['team_id']}): ").strip()

        nd = {}
        if new_name: nd["name"] = new_name
        if new_pos:  nd["pos"]  = require_position(new_pos)
        if new_age:  nd["age"]  = int(new_age)
        if new_price:nd["price"]= float(new_price)
        if new_team: nd["team_id"]= require_team_id(int(new_team))
        update_player(pid=pid, new_data=nd)
        print("✓ แก้ไข Player สำเร็จ")
    except Exception as e:
        print(f"❌ {e}")
```

```
elif choice == "3":
    pid = int(input("Player ID ที่จะลบ: ").strip())
    target = find_player(pid)
    if not target: print("❌ ไม่พบ Player ID นี้");  continue
    if not target["active"]: print("ℹ️ ผู้เล่นนี้ถูกลบอยู่แล้ว");  continue
    cf = input(f"ยืนยันลบ {target['name']} (ID {pid}) ? [y/N]: ").strip().lower()
    if cf == "y":
        update_player(pid=pid, new_data={"active": False})
        print("✓ ลบ Player (Soft Delete)")
    else:
        print("ยกเลิก")

elif choice == "4":
    teams = get_team_map()
    print("\nID   Name                  Pos Age  Price   Team (fullname)       Active")
    print("-------------------------------------------------------------------------")
    for p in read_players():
        tname = team_name_by_id(p["team_id"], teams)
        print(f"{p['id']:<4} {p['name']:<23} {p['pos']:<3} {p['age']:<3} {fmt_money(p['price']):>7}   {tname:<22} {p['active']}")

elif choice == "5":
    try:
        tid = int(input("Transfer ID: ").strip())
        ensure_unique_transfer_id(tid)
        pid = int(input("Player ID: ").strip())
        ply = find_player(pid)      You, 3 days ago • Project Com Pro …
        if not ply:             raise ValueError("ไม่พบ Player ID นี้")
        if not ply["active"]:   raise ValueError("ผู้เล่น inactive – ห้ามทำ Transfer")
        from_id = require_team_id(int(input("From Team ID: ").strip()))
        to_id   = require_team_id(int(input("To Team ID: ").strip()))
        if from_id == to_id:    raise ValueError("from/to ต้องต่างกัน")
        if ply["team_id"] != from_id:
            raise ValueError(f"from_team_id ต้องตรงกับทีมปัจจุบันของผู้เล่น (current={ply['team_id']})")
        age   = int(input("Age ตอนขาย: ").strip())
        price = float(input("Fee (ล้าน): ").strip())
        date  = int(input("Date (YYYYMMDD): ").strip())
        add_transfer(tid, pid, from_id, to_id, age, price, date, status=True, active=True)
        print("✓ เพิ่ม Transfer สำเร็จ")
    except Exception as e:
        print(f"❌ {e}")
elif choice == "6":
    rows = read_transfers()
    players = {p["id"]: p for p in read_players()}
    teams_map = get_team_map()
    print("\nID    Player_ID  Player Name         From (fullname)     To (fullname)       Age  Fee(M)  Pos    Status Active   Date")
    print("-----------------------------------------------------------------------------------------------------------------------")
    for t in rows:
        p = players.get(t["player_id"])
        pname = p["name"] if p else "-"
        pos   = (p["pos"] if p else "-").center(6)
        from_name = team_name_by_id(t["from"], teams_map)
        to_name   = team_name_by_id(t["to"], teams_map)
        print(f"{t['id']:<6}{t['player_id']:<11}{pname:<20}{from_name:<20}{to_name:<20}"
            f"{t['age']:<5}{fmt_money(t['price']):>7}  {pos}  {str(t['status']):<6} {str(t['active']):<6}  {t['date']}")

elif choice == "7":
    generate_report()
    print("✓ สร้างรายงาน summary.txt สำเร็จ")

elif choice == "0":
    print("👋 ออกโปรแกรมเรียบร้อย")
    break
else:
    print("❌ เมนูไม่ถูกต้อง ลองใหม่อีกครั้ง")
```

4.9 การจัดการข้อผิดพลาด

ในกรณีที่ผู้ใช้ป้อนข้อมูลไม่ถูกต้อง เช่น

- ป้อนค่าที่ **ไม่ใช่ตัวเลข** ในช่องที่ต้องกรอกเป็นตัวเลข เช่น Player ID, Age, Team ID
- หรือป้อนตัวเลขที่ **ไม่อยู่ในช่วงที่กำหนด** เช่น Team ID ไม่อยู่ระหว่าง 1–6 หรือ 99

เพื่อ **จับข้อผิดพลาด (ValueError)** ที่เกิดขึ้นจากการแปลงค่าหรือการตรวจสอบข้อมูล

เมื่อเกิดข้อผิดพลาด โปรแกรมจะไม่ค้างหรือหยุดทำงานทันที แต่จะแสดงข้อความแจ้งเตือนให้ผู้ใช้ทราบ

จากนั้นโปรแกรมจะ **ยุติการทำงานในรอบนั้น** และกลับไปยังเมนูหลัก เพื่อให้ผู้ใช้สามารถป้อนข้อมูลใหม่ได้อย่างถูกต้องในครั้งต่อไป

```
except Exception as e:
    print(f"❌ {e}")
```

4.10 การวนลูปกลับสู่เมนูหลัก

คำสั่ง while True:

ใช้สำหรับสร้างลูปที่ทำงานซ้ำไม่สิ้นสุด (Infinite Loop) เพื่อให้โปรแกรมคงอยู่ในหน้าเมนูหลักตลอดเวลา

คำสั่ง print()

ใช้สำหรับแสดงข้อความเมนูหลักของโปรแกรมให้ผู้ใช้เห็นทุกครั้งที่วนกลับมารอบใหม่

คำสั่ง input("เลือกเมนู: ")

ใช้สำหรับรับค่าหมายเลขเมนูที่ผู้ใช้เลือกจากแป้นพิมพ์

เมื่อผู้ใช้เลือกเมนู “0”

โปรแกรมจะทำงานในเงื่อนไข if choice == "0":

โดยแสดงข้อความ “ 👋 ออกโปรแกรมเรียบร้อย” แล้วใช้คำสั่ง break เพื่อออกจากลูป while True: และสิ้นสุดการทำงานของโปรแกรม

4.11 การออกจากโปรแกรม

เมื่อผู้ใช้เลือกหมายเลข “0” โปรแกรมจะพิมพ์ข้อความแจ้งออกจากระบบ และใช้คำสั่ง break เพื่อหยุดลูปหลักและจบการทำงานของโปรแกรมอย่างสมบูรณ์

```
elif choice == "0":
    print("👋 ออกโปรแกรมเรียบร้อย")
    break
```

สรุปการทำงานของฟังก์ชัน main()

ฟังก์ชัน main() (หรือส่วน if __name__ == "__main__":)

เป็นจุดเริ่มต้นของการทำงานของโปรแกรม โดยจะเรียกฟังก์ชัน main_menu() เพื่อเริ่มต้นระบบทั้งหมด

จากนั้นโปรแกรมจะรอรับคำสั่งจากผู้ใช้เพื่อจัดการข้อมูลในตลาดซื้อขายนักเตะ

และจะสิ้นสุดการทำงานเมื่อผู้ใช้เลือกเมนู “0” เพื่อออกจากโปรแกรม

ผลลัพธ์ฟังก์ชันเมน main()

ฟังก์ชั่น add_player(file):

ฟังก์ชันนี้ทำหน้าที่หลักในการ **สร้างและบันทึกรายการข้อมูลนักเตะใหม่** ภายในไฟล์ players.dat โดยจะรับข้อมูลจากผู้ใช้ เช่น **รหัสนักเตะ (Player ID), ชื่อ (Name), ตำแหน่ง (Position), อายุ (Age), ราคาค่าตัว (Price)** และ **ทีมต้นสังกัด (Team ID)** จากนั้นจะทำการบันทึกข้อมูลนักเตะทั้งหมดลงในไฟล์ในรูปแบบข้อมูลไบนารี

ฟังก์ชันนี้ถูกออกแบบให้สามารถตรวจสอบและป้องกันข้อผิดพลาดที่อาจเกิดขึ้นได้ เช่น การป้อนค่าที่ไม่ใช่ตัวเลข หรือข้อมูลที่เกินขนาดที่กำหนดไว้

โครงสร้างและการทำงานของฟังก์ชัน add_player(pid, name, pos, age, price, team_id, active=True, path="players.dat"):

ฟังก์ชันนี้อนุญาตให้ผู้ใช้สร้างข้อมูลนักเตะใหม่ในระบบ โดยป้อนค่าตามฟิลด์ที่กำหนด ได้แก่

- **pid** ⟶ รหัสนักเตะ (ต้องเป็นตัวเลขและไม่ซ้ำกับข้อมูลเดิม)
- **name** ⟶ ชื่อนักเตะ
- **pos** ⟶ ตำแหน่ง เช่น FW (กองหน้า), MF (กองกลาง), DF (กองหลัง), GK (ผู้รักษาประตู)
- **age** ⟶ อายุของนักเตะ
- **price** ⟶ ราคาค่าตัว (หน่วยล้านบาท)
- **team_id** ⟶ หมายเลขทีมต้นสังกัด (1–6 สำหรับทีมใหญ่ หรือ 99 สำหรับทีมอื่น ๆ)

ขั้นตอนการทำงานของฟังก์ชัน

1. **ระบบจะขอให้ผู้ใช้ป้อนข้อมูลนักเตะ**
   เช่น Player ID, Name, Position, Age, Price, Team ID
2. **ตรวจสอบความถูกต้องของข้อมูลที่ป้อน**
   ฟังก์ชันจะตรวจสอบประเภทข้อมูล เช่น อายุและราคาต้องเป็นตัวเลข และ Player ID ต้องไม่ซ้ำกับข้อมูลเดิมในไฟล์

3. **บันทึกข้อมูลลงในไฟล์ players.dat**

   ข้อมูลแต่ละรายการจะถูกจัดเก็บในรูปแบบ **Fixed-Length Record (ระเบียนความยาวคงที่)**

   โดยใช้โมดูล struct ผ่านตัวแปร PLAYER_STRUCT.pack() เพื่อบีบอัดข้อมูลให้มีขนาดคงที่ก่อนบันทึก

4. **เขียนข้อมูลลงไฟล์จริง**

   ใช้คำสั่ง f.write() เพื่อบันทึกข้อมูลนักเตะลงในไฟล์ในรูปแบบไบนารี ("ab")

   หากไม่มีไฟล์ players.dat ระบบจะสร้างไฟล์ใหม่ให้อัตโนมัติ

5. **แจ้งผู้ใช้เมื่อบันทึกสำเร็จ**

   หลังจากเขียนข้อมูลเสร็จ โปรแกรมจะพิมพ์ข้อความยืนยัน เช่น “✓ บันทึกผู้เล่นสำเร็จ” เพื่อให้ผู้ใช้ทราบว่าการเพิ่มข้อมูลสำเร็จแล้ว

โค้ดของฟังก์ชั่น

```
def add_player(pid: int, name: str, pos: str, age: int, price: float, team_id: int, active=True, path="players.dat"):
    with open(path, "ab") as f:
        f.write(PLAYER_STRUCT.pack(
            pid, pad_str(name, 40), pad_str(pos, 4), int(age), float(price), int(team_id), int(active)
        ))
```

การเปิดไฟล์ในโหมดสร้างข้อมูล(Write Mode)

ายในฟังก์ชันมีการเปิดไฟล์เพื่อเขียนข้อมูลใหม่ โดยใช้โหมด "wb" ซึ่งเป็นการเปิดไฟล์ใน **โหมดเขียนข้อมูลแบบไบนารี (Write Binary Mode)**

โหมด "w" หมายถึง **Write (เขียน)** ใช้สำหรับเขียนข้อมูลใหม่ลงในไฟล์ หากไฟล์นั้นมีอยู่แล้ว ข้อมูลเดิมทั้งหมดจะถูกลบและแทนที่ด้วยข้อมูลใหม่ที่เขียนเข้าไป ส่วน "b" หมายถึง **Binary (ไบนารี)** ใช้สำหรับเปิดไฟล์ในรูปแบบข้อมูลไบต์ เพื่อให้สามารถบันทึกข้อมูลในลักษณะตัวเลข ตัวอักษร หรือโครงสร้างข้อมูลแบบ struct ได้อย่างถูกต้อง

ภายในโค้ดจะใช้คำสั่ง with open(path, "rb") as f:

เพื่อเปิดไฟล์ในโหมดเขียนแบบไบนารี โดยใช้ **with** เพื่อจัดการการเปิดและปิดไฟล์อย่างปลอดภัย — เมื่อคำสั่งภายในบล็อก with ทำงานเสร็จ ระบบจะปิดไฟล์ให้อัตโนมัติทันที แม้เกิดข้อผิดพลาดระหว่างการทำงาน

```
with open(path, "rb") as f:
```

ฟังก์ชัน add_player()

ขั้นตอนการสร้างข้อมูลนักเตะใหม่ ฟังก์ชันจะเริ่มต้นด้วยการ **รับจำนวนรายการนักเตะที่ผู้ใช้ต้องการเพิ่มเข้าสู่ระบบ** ผ่านการป้อนค่าทางคีย์บอร์ด โดยโปรแกรมจะขอให้ผู้ใช้ระบุจำนวนข้อมูลที่ต้องการบันทึก เช่น จำนวนผู้เล่นที่ต้องการเพิ่มทั้งหมด จากนั้นค่าที่ผู้ใช้ป้อนจะถูกเก็บไว้ในตัวแปร count ซึ่งเป็นจำนวนครั้งที่ระบบจะวนลูปเพื่อรับข้อมูลของนักเตะแต่ละคน

หากผู้ใช้ป้อนค่าที่ไม่ใช่ตัวเลข เช่น ตัวอักษรหรือช่องว่าง โปรแกรมจะเกิดข้อผิดพลาดประเภท **ValueError** เนื่องจากไม่สามารถแปลงข้อมูลที่ป้อนได้เป็นชนิดตัวเลขได้ ฟังก์ชันจะแสดงข้อความแจ้งเตือนเพื่อแนะนำให้ผู้ใช้ป้อนข้อมูลที่ถูกต้องอีกครั้ง

```python
def add_player(pid: int, name: str, pos: str, age: int, price: float, team_id: int, active=True, path="players.dat"):
    with open(path, "ab") as f:
        f.write(PLAYER_STRUCT.pack(
            pid, pad_str(name, 40), pad_str(pos, 4), int(age), float(price), int(team_id), int(active)
        ))
```

ฟังก์ชัน update_player()

ฟังก์ชันนี้ทำหน้าที่หลักในการ **แก้ไขข้อมูลของผู้เล่นในระบบ** ที่ถูกบันทึกไว้ภายในไฟล์ players.dat โดยผู้ใช้จะระบุรหัสผู้เล่น (Player ID) ที่ต้องการแก้ไข จากนั้นระบบจะค้นหาผู้เล่นในไฟล์และอนุญาตให้แก้ไขข้อมูล เช่น ชื่อ ตำแหน่ง อายุ ราคาค่าตัว และทีมต้นสังกัด

หากพบรหัสผู้เล่นที่ตรงกับข้อมูลในไฟล์ ระบบจะทำการอัปเดตข้อมูลใหม่และเขียนทับข้อมูลเดิมในไฟล์ หากไม่พบรหัสผู้เล่นที่ตรงกัน ฟังก์ชันจะแสดงข้อความแจ้งเตือนว่าไม่พบข้อมูลผู้เล่นตามที่ระบุไว้

```python
def update_player(path="players.dat", pid=None, new_data=None):
    if not os.path.exists(path): return
    with open(path, "r+b") as f:
        idx = 0
        while True:
            b = f.read(PLAYER_STRUCT.size)
            if not b or len(b) < PLAYER_STRUCT.size: break
            cur = PLAYER_STRUCT.unpack(b)
            if cur[0] == pid:
                # cur = (id, name(bytes), pos(bytes), age, price, team_id, active)
                name = new_data.get("name", read_str(cur[1]))
                pos  = new_data.get("pos",  read_str(cur[2]))
                age  = new_data.get("age",  cur[3])
                price= new_data.get("price",cur[4])
                team = new_data.get("team_id", cur[5])
                active = int(new_data.get("active", cur[6]))
                f.seek(idx * PLAYER_STRUCT.size)
                f.write(PLAYER_STRUCT.pack(
                    cur[0], pad_str(name,40), pad_str(pos,4), int(age), float(price), int(team), active
                ))
                return
            idx += 1
```

ฟังก์ชัน add_transfer()

ฟังก์ชันนี้ทำหน้าที่หลักในการ **บันทึกข้อมูลการย้ายทีมของนักเตะ (Transfer Record)** ลงในไฟล์ transfers.dat โดยผู้ใช้จะต้องระบุข้อมูลการย้ายทีม เช่น **รหัสการย้ายทีม (Transfer ID)**, **รหัสนักเตะ (Player ID)**, **ทีมต้นสังกัดเดิม**, **ทีมใหม่**, **ราคาการย้ายทีม**, และ **วันที่ย้ายทีม**

ฟังก์ชันนี้ถูกออกแบบให้สามารถตรวจสอบความถูกต้องของข้อมูลก่อนบันทึก เพื่อป้องกันการซ้ำซ้อนของรหัส และรับประกันว่าข้อมูลแต่ละรายการจะถูกจัดเก็บในรูปแบบไบนารี (Binary Format) อย่างถูกต้องตามโครงสร้างที่กำหนด

```python
def add_transfer(tid:int, pid:int, from_id:int, to_id:int, age:int, price:float,
                 date:int, status=True, active=True, path="transfers.dat", player_file="players.dat"):
    # เขียนดีล
    with open(path, "ab") as f:
        f.write(TRANSFER_STRUCT.pack(
            int(tid), int(pid), int(from_id), int(to_id), int(age), float(price), int(date), int(status), int(active)
        ))

    # อัปเดตทีมของผู้เล่น -> current_team_id = to_id
    if os.path.exists(player_file):
        with open(player_file, "r+b") as pf:
            idx = 0
            while True:
                b = pf.read(PLAYER_STRUCT.size)
                if not b or len(b) < PLAYER_STRUCT.size: break
                rec = PLAYER_STRUCT.unpack(b)
                if rec[0] == pid:
                    updated = (rec[0], rec[1], rec[2], rec[3], rec[4], int(to_id), rec[6])
                    pf.seek(idx * PLAYER_STRUCT.size)
                    pf.write(PLAYER_STRUCT.pack(*updated))
                    break
                idx += 1
```

ฟังก์ชัน read_players()

ฟังก์ชันนี้ทำหน้าที่หลักในการ **อ่านข้อมูลของผู้เล่นทั้งหมดที่บันทึกไว้ในไฟล์ players.dat** เพื่อนำมาแสดงผลหรือใช้ประมวลผลในส่วนอื่นของระบบ เช่น การแก้ไขข้อมูล การแสดงรายงาน หรือการย้ายทีมข้อมูลที่ถูกบันทึกในไฟล์จะอยู่ในรูปแบบ **ข้อมูลไบนารี (Binary Format)** ซึ่งมี

โครงสร้างคงที่ (Fixed-Length Record) ดังนั้น ฟังก์ชันนี้จะต้องใช้คำสั่ง struct.unpack() เพื่อถอดรหัสข้อมูลแต่ละระเบียนให้อยู่ในรูปแบบที่สามารถนำมาใช้งานได้ เช่น dictionary

```
def read_players(path="players.dat"):
    rows = []
    if not os.path.exists(path): return rows
    with open(path, "rb") as f:
        while True:
            b = f.read(PLAYER_STRUCT.size)
            if not b or len(b) < PLAYER_STRUCT.size: break
            pid, name, pos, age, price, team_id, active = PLAYER_STRUCT.unpack(b)
            rows.append({
                "id": pid, "name": read_str(name), "pos": read_str(pos), "age": age,
                "price": float(price), "team_id": team_id, "active": bool(active)
            })
    return rows
```

ฟังก์ชัน read_transfers()

ฟังก์ชันนี้ทำหน้าที่หลักในการ **อ่านข้อมูลการย้ายทีมของนักเตะทั้งหมด** ที่ถูกบันทึกไว้ในไฟล์ transfers.dat เพื่อนำมาแสดงผลหรือใช้ในการประมวลผลในส่วนอื่นของโปรแกรม เช่น การสร้างรายงานสรุปการซื้อขาย หรือการตรวจสอบประวัติการย้ายทีมของผู้เล่นแต่ละคน

ข้อมูลทั้งหมดในไฟล์จะถูกเก็บในรูปแบบ **ข้อมูลไบนารี (Binary Format)** ซึ่งมีขนาดคงที่ (Fixed-Length Record) ดังนั้น ฟังก์ชันนี้ต้องใช้คำสั่ง struct.unpack() เพื่อถอดรหัสข้อมูลให้อยู่ในรูปแบบที่สามารถอ่านและนำไปใช้งานได้

```
def read_transfers(path="transfers.dat"):
    rows = []
    if not os.path.exists(path): return rows
    with open(path, "rb") as f:
        while True:
            b = f.read(TRANSFER_STRUCT.size)
            if not b or len(b) < TRANSFER_STRUCT.size: break
            tid, pid, from_id, to_id, age, price, date, status, active = TRANSFER_STRUCT.unpack(b)
            rows.append({
                "id": tid, "player_id": pid, "from": from_id, "to": to_id,
                "age": age, "price": float(price), "date": date,
                "status": bool(status), "active": bool(active)
            })
    return rows
```

ฟังก์ชัน generate_report()

```
def generate_report(team_file="teams.dat", player_file="players.dat",
                    transfer_file="transfers.dat", report_file="summary.txt"):
    teams_map   = get_team_map()
    players     = read_players()
    players_map = {p["id"]: p for p in players}
    transfers   = read_transfers()
```

กำหนดตัวแปร

```
DEFAULT_TEAM_NAMES = {
    1:"Arsenal", 2:"Chelsea", 3:"Liverpool",
    4:"Manchester City", 5:"Manchester United",
    6:"Tottenham Hotspur", 99:"Other Team"
}
```

กำหนดชื่อทีมพื้นฐาน

```
def name_by_id(tid: int) -> str:
    return teams_map.get(tid, {}).get("name") or DEFAULT_TEAM_NAMES.get(tid, str(tid))
```

คืนค่าชื่อทีมตามรหัสที่ระบุ

```
HDRS   = ["TID","Player_ID","Player","from_Team","to_Team","Age","Fee(M)","Position","Status","Active","Date"]
WIDTHS = [   5,        10,      20,        18,       18,    5,     8,        9,        8,       8,     12]
```

กำหนด Layout ของตาราง

```
def _sep():
    return "+" + "+".join("-"*(w+2) for w in WIDTHS) + "+\n"
```

เส้นคั่นตามความกว้างจริงของแต่ละคอลัมน์

```
def _hdr():
    return "| " + " | ".join(f"{h:^{w}}" for h, w in zip(HDRS, WIDTHS)) + " |\n"
```

จัดแถวหัวคอลัมน์ให้อยู่กึ่งกลาง

```
with open(report_file, "w", encoding="utf-8") as f:
```

เปิดไฟล์ ในโหมดเขียน (Write Mode)

```
f.write("Big 6 Transfer Market - (Summary Report)\n")
f.write(f"Generated at : {dt.datetime.now():%Y-%m-%d %H:%M:%S} (+07:00)\n")
f.write("App version  : 1.0\n")
f.write("Endianness   : Little-Endian\n")
f.write("Encoding     : UTF-8 (Fixed-Length)\n\n")
```

เขียนส่วนหัวของรายงาน (Report Header)

```
f.write("+----------------+\n| Transfer Table |\n+----------------+\n\n")
f.write(_sep())
f.write(_hdr())
f.write(_sep())
```

สร้างและจัดรูปแบบหัวตาราง

```
for t in transfers:
    p = players_map.get(t["player_id"])
    pname = (p["name"] if p else "-")[:WIDTHS[2]]
    pos   = (p["pos"] if p else "-")
    from_name = name_by_id(t["from"])[:WIDTHS[3]]
    to_name   = name_by_id(t["to"])[:WIDTHS[4]]
    dstr = yyyymmdd_to_str(t["date"])

    cells = [
        f"{t['id']:<{WIDTHS[0]}}",
        f"{t['player_id']:<{WIDTHS[1]}}",
        f"{pname:<{WIDTHS[2]}}",
        f"{from_name:<{WIDTHS[3]}}",
        f"{to_name:<{WIDTHS[4]}}",
        f"{t['age']:<{WIDTHS[5]}}",
        f"{t['price']:{WIDTHS[6]}.1f}",
        f"{pos:^{WIDTHS[7]}}",
        f"{str(t['status']):<{WIDTHS[8]}}",
        f"{str(t['active']):<{WIDTHS[9]}}",
        f"{dstr:<{WIDTHS[10]}}",
    ]
    f.write("| " + " | ".join(cells) + " |\n")
```

เขียนข้อมูลการย้ายทีมแต่ละรายการของผู้เล่นลงในรายงาน

```
f.write(_sep())
f.write("\n")
```

สร้างและจัดรูปแบบท้ายตาราง

```
act_players = [p for p in players if p["active"]]
vals = [p["price"] for p in act_players]
f.write("Summary (Active Player)\n")
f.write(f"- Total Player : {len(players)}\n")
f.write(f"- Active Players: {len(act_players)}\n")
f.write(f"- Deleted Player: {len(players) - len(act_players)}\n")
if vals:
    f.write(f"- Average ValueM: {sum(vals)/len(vals):.1f}\n")
    f.write(f"- Max ValueM : {max(vals):.1f}\n")
    f.write(f"- Min ValueM : {min(vals):.1f}\n")
f.write("\n")
```

สรุปจำนวนผู้เล่นและมูลค่าของนักเตะในระบบ

```
act_trans = [t for t in transfers if t["active"]]
fees = [t["price"] for t in act_trans]
f.write("Transfer (Active Only)\n")
f.write(f"- Total Transfer: {len(act_trans)}\n")
if fees:
    f.write(f"- Highest Fee : {max(fees):.1f}\n")
    f.write(f"- Lowest Fee  : {min(fees):.1f}\n")
    f.write(f"- Average Fee : {sum(fees)/len(fees):.1f}\n")
```

สรุปข้อมูลการย้ายทีมที่ยัง Active อยู่ในระบบ

```python
def main_menu():
    while True:
        print("\n================ BIG 6 TRANSFER MARKET ================")
        print("1) Add Player")
        print("2) Update Player")
        print("3) Delete Player")
        print("4) View Players")
        print("5) Add Transfer")
        print("6) View Transfers")
        print("7) Generate Report")
        print("0) Exit")
        print("=======================================================")

        choice = input("เลือกเมนู: ").strip()
```

แสดงเมนูหลักของระบบ Big 6 Transfer Market

```python
if choice == "1":
    try:
        pid = int(input("Player ID: ").strip())
        ensure_unique_player_id(pid)
        name = input("Name: ").strip()
        pos  = require_position(input("Position (FW/MF/DF/GK/...): ").strip())
        age  = int(input("Age: ").strip())
        price= float(input("Price Buy (ล้าน): ").strip())
        team = require_team_id(int(input("Team ID (1-6 หรือ 99=OTH): ").strip()))
        add_player(pid, name, pos, age, price, team, active=True)
        print("✓ เพิ่ม Player สำเร็จ")
    except Exception as e:
        print(f"❌ {e}")
```

เมนูที่ 1 เพิ่มรายชื่อนักเตะ

```python
elif choice == "2":
    try:
        pid = int(input("Player ID ที่จะแก้ไข: ").strip())
        cur = find_player(pid)
        if not cur:
            print("❌ ไม่พบ Player ID นี้");  continue
        new_name = input(f"Name ใหม่ (Enter = {cur['name']}): ").strip()
        new_pos  = input(f"Position ใหม่ (Enter = {cur['pos']}): ").strip()
        new_age  = input(f"Age ใหม่ (Enter = {cur['age']}): ").strip()
        new_price= input(f"Price ใหม่ (Enter = {fmt_money(cur['price'])}): ").strip()
        new_team = input(f"Team ID ใหม่ (Enter = {cur['team_id']}): ").strip()

        nd = {}
        if new_name: nd["name"] = new_name
        if new_pos:  nd["pos"]  = require_position(new_pos)
        if new_age:  nd["age"]  = int(new_age)
        if new_price:nd["price"]= float(new_price)
        if new_team: nd["team_id"]= require_team_id(int(new_team))
        update_player(pid=pid, new_data=nd)
        print("✓ แก้ไข Player สำเร็จ")
    except Exception as e:
        print(f"❌ {e}")
```

เมนูที่ 2 อัพเดทและแก้ไขข้อมูลนักเตะ

```
elif choice == "3":
    pid = int(input("Player ID ที่จะลบ: ").strip())
    target = find_player(pid)
    if not target: print("❌ ไม่พบ Player ID นี้");  continue
    if not target["active"]: print("ℹ️ ผู้เล่นนี้ถูกลบอยู่แล้ว");  continue
    cf = input(f"ยืนยันลบ {target['name']} (ID {pid}) ? [y/N]: ").strip().lower()
    if cf == "y":
        update_player(pid=pid, new_data={"active": False})
        print("✓ ลบ Player (Soft Delete)")
    else:
        print("ยกเลิก")
```

เมนูที่ 3 ลบรายชื่อนักเตะ

```
elif choice == "4":
    teams = get_team_map()
    print("\nID   Name                 Pos Age  Price   Team (fullname)       Active")
    print("-------------------------------------------------------------------------------")
    for p in read_players():
        tname = team_name_by_id(p["team_id"], teams)
        print(f"{p['id']:<4} {p['name']:<23} {p['pos']:<3} {p['age']:<3} {fmt_money(p['price']):>7}   {tname:<22} {p['active']}")
```

เมนูที่ 4 แสดงรายชื่อนักเตะที่บันทึกไว้

```
elif choice == "5":
    try:
        tid = int(input("Transfer ID: ").strip())
        ensure_unique_transfer_id(tid)
        pid = int(input("Player ID: ").strip())
        ply = find_player(pid)
        if not ply:             raise ValueError("ไม่พบ Player ID นี้")
        if not ply["active"]:   raise ValueError("ผู้เล่น inactive – ห้ามทำ Transfer")
        from_id = require_team_id(int(input("From Team ID: ").strip()))
        to_id   = require_team_id(int(input("To Team ID: ").strip()))
        if from_id == to_id:    raise ValueError("from/to ต้องต่างกัน")
        if ply["team_id"] != from_id:
            raise ValueError(f"from_team_id ต้องตรงกับทีมปัจจุบันของผู้เล่น (current={ply['team_id']})")
        age   = int(input("Age ตอนขาย: ").strip())
        price = float(input("Fee (ล้าน): ").strip())
        date  = int(input("Date (YYYYMMDD): ").strip())
        add_transfer(tid, pid, from_id, to_id, age, price, date, status=True, active=True)
        print("✓ เพิ่ม Transfer สำเร็จ")
    except Exception as e:
        print(f"❌ {e}")
```

เมนูที่ 5 บันทึกข้อมูลการซื้อ-ขายนักเตะ

```
elif choice == "6":
    rows = read_transfers()
    players = {p["id"]: p for p in read_players()}
    teams_map = get_team_map()
    print("\nID    Player_ID  Player Name         From (fullname)     To (fullname)       Age  Fee(M)  Pos    Status Active   Date")
    print("-------------------------------------------------------------------------------------------------------------------------")
    for t in rows:
        p = players.get(t["player_id"])
        pname = p["name"] if p else "-"
        pos   = (p["pos"] if p else "-").center(6)
        from_name = team_name_by_id(t["from"], teams_map)
        to_name   = team_name_by_id(t["to"], teams_map)
        print(f"{t['id']:<6}{t['player_id']:<11}{pname:<20}{from_name:<20}{to_name:<20}"
            f"{t['age']:<5}{fmt_money(t['price']):>7}  {pos}  {str(t['status']):<6} {str(t['active']):<6}  {t['date']}")
```

เมนูที่ 6 แสดงบันทึกข้อมูลการซื้อขายนักเตะ

```
elif choice == "7":
    generate_report()
    print("✓ สร้างรายงาน summary.txt สำเร็จ")
```

เมนูที่ 7 สร้างรายงานสรุปผลการซื้อ-ขายนักเตะ

```
elif choice == "0":
    print("👋 ออกโปรแกรมเรียบร้อย")
    break
```

เมนูที่ 8 ออกจากโปรแกรม

```
else:
    print("❌ เมนูไม่ถูกต้อง ลองใหม่อีกครั้ง")
```

แจ้งเตือนข้อผิดพลาดการเลือกเมนู

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

เริ่มการทำงานของโปรแกรม