ส่วนที่ 1 Core Of DB:

Full Source From : db_connector.py

1. การกำหนดค่า สำหรับการเชื่อมต่อ DB

```
# ================== SQL Server Configuration ==================
# **** โปรดแก้ไขค่าเหล่านี้ให้ตรงกับเครื่องของคุณ ****
server = 'DESKTOP-1UPQK9B\\SQLEXPRESS'
database = 'BusTicketSystem'
driver = '{ODBC Driver 17 for SQL Server}'
```

ตัวแปรตั้งค่า (Configuration Variables):

- server = 'DESKTOP-1UPQK9B\\SQLEXPRESS' : ระบุ ชื่อเซิร์ฟเวอร์ และชื่อ Instance ของ SQL Server ที่ติดตั้งอยู่
- database = 'BusTicketSystem' : ระบุ ชื่อฐานข้อมูล ที่ต้องการเชื่อมต่อ
- driver = '{ODBC Driver 17 for SQL Server}' : ระบุ ไดรเวอร์ ODBC ที่ใช้ในการสื่อสารกับฐานข้อมูล

2. ฟังก์ชันสำหรับการเชื่อมต่อ DB : connect_db()

```python
def connect_db():
    conn_str = (
        f'DRIVER={driver};'
        f'SERVER={server};'
        f'DATABASE={database};'
        f'Trusted_Connection=yes;'
    )
    try:
        conn = pyodbc.connect(conn_str)
        return conn
    except Exception as e:
        temp_root = Tk()
        temp_root.withdraw()
        messagebox.showerror("Database Error", f"Cannot connect to SQL Server: {e}")
        temp_root.destroy()
        return None
```

- สร้าง สตริงการเชื่อมต่อ (conn_str) โดยใช้ค่าจากตัวแปรข้างต้น
- ใช้ pyodbc.connect(conn_str) เพื่อพยายามสร้าง การเชื่อมต่อ (Connection object) กับ SQL Server
- ใช้ Trusted_Connection=yes ซึ่งหมายถึงการเชื่อมต่อโดยใช้ Windows Authentication (ไม่จำเป็นต้องใช้ Username/Password ของ SQL Server)
- หากเชื่อมต่อ สำเร็จ จะคืนค่า Connection object
- หากเชื่อมต่อ ล้มเหลว จะแสดง กล่องข้อความแจ้งเตือนข้อผิดพลาด (Database Error) โดยใช้ tkinter.messagebox.showerror และคืนค่า None

3.การตรวจสอบสิทธิ์ผู้ใช้ (User Authentication)

```python
def authenticate_user(username, password):
    conn = connect_db()
    if conn:
        cursor = conn.cursor()
        try:
            query = "SELECT StaffID, StaffName, Role FROM STAFF WHERE Username = ? AND Password = ?"
            cursor.execute(query, (username, password))
            user_data = cursor.fetchone()
            conn.close()
            return user_data
        except Exception as e:
            messagebox.showerror("Query Error", f"Error during authentication: {e}")
            conn.close()
            return None
    return None
```

ฟังก์ชัน `authenticate_user(username, password)`:

- เรียกใช้ `connect_db()` เพื่อสร้างการเชื่อมต่อ
- หากเชื่อมต่อสำเร็จ จะสร้าง Cursor object เพื่อใช้ในการดำเนินการกับฐานข้อมูล
- การทำงานกับ SQL:
    - รันคำสั่ง SQL SELECT เพื่อดึงข้อมูล `StaffID`, `StaffName`, และ `Role` จากตาราง STAFF
    - เงื่อนไขคือ `WHERE Username = ? AND Password = ?` โดยที่ ? จะถูกแทนที่ด้วยค่า `username` และ `password` ที่รับเข้ามา (เป็นการป้องกัน SQL Injection)
    - ใช้ `cursor.fetchone()` เพื่อดึงผลลัพธ์ หนึ่งแถวแรก ที่ตรงกับเงื่อนไข
- หากตรวจสอบสิทธิ์ สำเร็จ จะคืนค่าข้อมูลพนักงาน (`user_data`)
- หาก ไม่พบ พนักงานที่ตรงกัน จะคืนค่า None
- มีการจัดการข้อผิดพลาดในการรัน Query และจะปิดการเชื่อมต่อทุกครั้ง

4. การดึงข้อมูลผู้ใช้ที่เข้าสู่ระบบ (Fetching Logged-in User Information)

```python
def get_logged_in_user_info(staff_id):
    conn = connect_db()
    if conn:
        cursor = conn.cursor()
        try:
            cursor.execute("SELECT StaffName, Role FROM STAFF WHERE StaffID = ?", (staff_id,))
            user_info = cursor.fetchone()
            conn.close()
            return user_info
        except Exception as e:
            messagebox.showerror("Query Error", f"Error fetching user info: {e}")
            conn.close()
            return None
    return None
```

ฟังก์ชัน `get_logged_in_user_info(staff_id)`:

- เรียกใช้ `connect_db()` เพื่อสร้างการเชื่อมต่อ
- หากเชื่อมต่อสำเร็จ จะสร้าง Cursor object
- การทำงานกับ SQL:
    - รันคำสั่ง SQL SELECT เพื่อดึงข้อมูล `StaffName` และ `Role` จากตาราง STAFF
    - เงื่อนไขคือ `WHERE StaffID = ?` โดยที่ ? ถูกแทนที่ด้วยค่า `staff_id` ที่รับเข้ามา
- ใช้ `cursor.fetchone()` เพื่อดึงผลลัพธ์
- หากดึงข้อมูล สำเร็จ จะคืนค่าข้อมูลชื่อและบทบาทของพนักงาน (`user_info`)
- หากดึงข้อมูล ล้มเหลว จะคืนค่า None
- มีการจัดการข้อผิดพลาดและปิดการเชื่อมต่อทุกครั้ง

ส่วนที่ 2 Login

Full Code From : login.py

1. Import ฟังก์ชั่น authenticate_user จาก db_connector

```
import tkinter as tk
from tkinter import messagebox
from db_connector import authenticate_user
import Bus_Select_gui
```

authenticate_user จะ:

- สร้างการเชื่อมต่อกับ SQL Server (ไปยังฐานข้อมูล BusTicketSystem).
- รันคำสั่ง SQL: SELECT StaffID, StaffName, Role FROM STAFF WHERE Username = ? AND Password = ?
- ถ้าพบข้อมูลที่ตรงกัน จะคืนค่า (StaffID, StaffName, Role) กลับมา
- ถ้าไม่พบหรือมีข้อผิดพลาด จะคืนค่า None

2. ขั้นตอนการตรวจสอบสิทธิ์

```
def login_attempt():
    username = user_entry.get()
    password = pass_entry.get()

    if not username or not password:
        messagebox.showwarning("Warning", "กรุณากรอก Username และ Password ให้ครบถ้วน")
        return

    user_info = authenticate_user(username, password)

    if user_info:
        staff_id, staff_name, staff_role = user_info
        messagebox.showinfo("Success", f"Login สำเร็จ! ยินดีต้อนรับคุณ {staff_name}")

        Bus_Select_gui.open_select_bus_screen(login_root, staff_id)

    else:
        messagebox.showerror("Error", "Username หรือ Password ไม่ถูกต้อง!")
```

การทำงานจริงที่เชื่อมต่อกับ SQL Server อยู่ภายในฟังก์ชัน **`login_attempt()`**

ดึงข้อมูลอินพุต :

username = user_entry.get()

password = pass_entry.get()

โปรแกรมดึงค่า Username และ Password ที่ผู้ใช้ป้อนจากช่องป้อนข้อมูล (tk.Entry)

เรียกใช้ฟังก์ชันตรวจสอบสิทธิ์ (SQL Interaction):
user_info = authenticate_user(username, password)

ส่วนนี้คือจุดที่โปรแกรมติดต่อกับ SQL Server โดยการเรียก authenticate_user() พร้อมส่ง username และ password เป็นพารามิเตอร์ เพื่อให้ฟังก์ชันไปตรวจสอบกับตาราง STAFF ในฐานข้อมูล

ประมวลผลจาก SQL:

- **กรณีที่ 1: Login สำเร็จ (if user_info:)**
    - หมายความว่าฐานข้อมูล พบ พนักงานที่ตรงกับ Username และ Password ที่ป้อนเข้ามา
    - ค่าที่ได้คืนมา (user_info) จะถูกแยกเป็น staff_id, staff_name, และ staff_role
    - แสดง messagebox.showinfo("Success", ...)
    - เรียกหน้าจอถัดไป Bus_Select_gui.open_select_bus_screen(...) พร้อมส่ง staff_id ที่ได้มาจากฐานข้อมูล เพื่อนำไปใช้ในการดำเนินงานอื่น ๆ ต่อไปในระบบ
- **กรณีที่ 2: Login ล้มเหลว (else:)**
    - หมายความว่าฐานข้อมูล ไม่พบ พนักงานที่ตรงกับข้อมูลที่ป้อน
    - แสดง messagebox.showerror("Error", ...)

ส่วนที่ 3 : การเลือก รถเมล์สายที่พนักงานทำงาน และ ราคาตั๋วที่กำหนด

Full Source From: Bus_Select_gui.py

โค้ดใช้ฟังก์ชัน `get_logged_in_user_info()` ที่นำเข้าจากโมดูล `db_connector` เพื่อดึงชื่อและบทบาทของพนักงาน

```
import tkinter as tk
from tkinter import messagebox
from db_connector import get_logged_in_user_info
import login
import ticket_price_gui
```

การเรียกใช้ฟังก์ชัน :

```
    # ดึงข้อมูลผู้ใช้ (ชื่อและ Role)
    user_data = get_logged_in_user_info(staff_id)
    staff_name = user_data[0] if user_data else "Guest"
    staff_role = user_data[1] if user_data else "Unknown"
```

- เมื่อฟังก์ชัน open_select_bus_screen ถูกเรียก จะได้รับค่า staff_id ที่ส่งต่อมาจากหน้า Login
- ฟังก์ชันจะเรียก get_logged_in_user_info() โดยส่ง staff_id ไป
- db_connector จะใช้ staff_id นี้ไปรันคำสั่ง SQL SELECT StaffName, Role FROM STAFF WHERE StaffID = ?
- ฐานข้อมูลจะส่งค่า staff_name และ staff_role กลับมาในตัวแปร user_data

ข้อมูลที่ได้จากฐานข้อมูลจะถูกนำมาใช้เพื่อ:

- แสดง ชื่อพนักงาน บนข้อความต้อนรับ (e.g., "welcome Khun [staff_name]").
- ตรวจสอบ บทบาท (`staff_role`) เพื่อแสดงปุ่ม "Admin Console" หากพนักงานคนนั้นมีบทบาทเป็น 'Admin'

การดึงข้อมูลรายการราคาตั๋ว (Fetching Fare Rates)

Full Source From : ticket_price_gui.py

- **คำสั่ง Import และเรียกใช้:**

```
import tkinter as tk
from tkinter import messagebox

from db_connector import get_fare_rates, connect_db
import Bus_Select_gui
import ticket_print
from datetime import datetime # ต้อง Import เพื่อใช้สร้าง TicketNo
```

ฟังก์ชัน open_select_price_screen จะเรียกใช้ get_fare_rates() เพื่อดึงรายการราคาตั๋วที่กำหนดไว้มาเก็บไว้ในตัวแปร prices
การทำงานกับ SQL ใน db_connector: ฟังก์ชันนี้ในโมดูล db_connector.py จะรันคำสั่ง SQL เพื่อ SELECT รายการราคาที่ใช้ได้จากตารางใดตารางหนึ่งในฐานข้อมูลจากตาราง FARE_RATE และส่งผลลัพธ์เป็นลิสต์ของราคา พื่อนำมาสร้างเป็นปุ่มบนหน้าจอ

- **การบันทึกข้อมูลตั๋วใหม่ลงในฐานข้อมูล (Inserting New Ticket Record)**

ขั้นตอนนี้เกิดขึ้นภายในฟังก์ชัน `confirm_ticket(price, root, staff_id, bus_no)` เมื่อผู้ใช้กดปุ่มยืนยันการจำหน่ายตั๋ว

```
def confirm_ticket(price, root, staff_id, bus_no):
    """แสดง Pop-up ยืนยันราคา และบันทึกข้อมูลตั๋วลง DB ก่อนไปหน้าพิมพ์"""

    answer = messagebox.askyesno("Confirm this ticket?", f"Ticket Price: {price} THB\nConfirm purchase?")

    if answer:
        conn = connect_db()
        if conn:
            try:
                # 1. สร้าง Ticket No ที่ไม่ซ้ำกัน (ใช้ Timestamp)
                current_time = datetime.now()
                ticket_no = f"TKT{bus_no}-{current_time.strftime('%Y%m%d%H%M%S')}"

                # 2. กำหนด RouteID ชั่วคราว (ใช้ค่า 1 เป็นตัวอย่าง)
                route_id = 1

                cursor = conn.cursor()

                # 3. คำสั่ง INSERT ข้อมูลลงตาราง TICKET
                # Fields: TicketNo (PK), BusID, StaffID, Price, RouteID, Datetime, TicketStatus
                query = """
                INSERT INTO TICKET (TicketNo, BusID, StaffID, Price, RouteID, Datetime, TicketStatus)
                VALUES (?, ?, ?, ?, ?, ?, 'Active')
                """

                cursor.execute(query,
                               ticket_no,
                               bus_no,
                               staff_id,
                               price,
                               route_id,
                               current_time)

                conn.commit()
                conn.close()

                messagebox.showinfo("Success", f"Ticket {ticket_no} confirmed and saved to DB!")

                # 4. เมื่อบันทึกสำเร็จ ไปหน้า Ticket Print
                ticket_print.open_ticket_print_screen(root, staff_id, bus_no, price)
                return

            except Exception as e:
                if conn: conn.close()
                messagebox.showerror("DB Error", f"Failed to save ticket: {e}")
                return # ไม่ไปหน้าพิมพ์ถ้า DB Error

        else:
            messagebox.showerror("DB Connection", "Could not connect to database to save ticket.")

    else:
        messagebox.showinfo("Cancel", "Purchase Canceled")
```

การเชื่อมต่อฐานข้อมูล

- โปรแกรมเรียกใช้ `connect_db()` เพื่อสร้างการเชื่อมต่อกับ SQL Server (conn = connect_db())

การเตรียมข้อมูล

- สร้าง TicketNo

```
ticket_no = f"TKT{bus_no}-{current_time.strftime('%Y%m%d%H%M%S')}"
```

- สร้างหมายเลขตั๋วที่ไม่ซ้ำกัน (PK) โดยใช้หมายเลขรถ (bus_no) และเวลาปัจจุบัน (Timestamp)

คำสั่ง SQL INSERT (การบันทึกข้อมูล)

```
        query = """
        INSERT INTO TICKET (TicketNo, BusID, StaffID, Price, RouteID, Datetime, TicketStatus)
        VALUES (?, ?, ?, ?, ?, ?, 'Active')
        """

        cursor.execute(query,
                       ticket_no,
                       bus_no,
                       staff_id,
                       price,
                       route_id,
                       current_time)

        conn.commit()
        conn.close()

        messagebox.showinfo("Success", f"Ticket {ticket_no} confirmed and saved to DB!")

        # 4. เมื่อบันทึกสำเร็จ ไปหน้า Ticket Print
        ticket_print.open_ticket_print_screen(root, staff_id, bus_no, price)
        return

    except Exception as e:
        if conn: conn.close()
```

การดำเนินการ: โค้ดสร้าง Cursor และรันคำสั่ง INSERT เพื่อบันทึกข้อมูลการขายตั๋วใหม่ลงในตาราง TICKET
ข้อมูลที่ถูกบันทึก:

- TicketNo (รหัสตั๋วที่สร้างขึ้น)
- BusID (หมายเลขรถที่ได้รับมาจากหน้าจอก่อนหน้า)
- StaffID (รหัสพนักงานที่เข้าสู่ระบบ)
- Price (ราคาที่ผู้ใช้เลือก)
- RouteID (ค่าตัวอย่าง 1)
- Datetime (เวลาที่บันทึกรายการ)
- TicketStatus (ตั้งค่าเป็น 'Active')

ยืนยันการเปลี่ยนแปลงและการปิดการเชื่อมต่อ

- conn.commit(): เป็นคำสั่งสำคัญที่ส่งข้อมูลที่รันคำสั่ง INSERT ไป บันทึกถาวร ในฐานข้อมูล SQL Server
- conn.close(): ปิดการเชื่อมต่อเพื่อคืนทรัพยากร

ถ้าการบันทึกสำเร็จ โปรแกรมจะแสดง messagebox.showinfo("Success", ...) และนำไปสู่หน้าจอพิมพ์ตั๋ว (ticket_print)

หากมีข้อผิดพลาดในการเชื่อมต่อหรือการรัน Query (เช่น Exception as e) โปรแกรมจะแสดงข้อความแจ้งเตือนข้อผิดพลาด (messagebox.showerror) แทน และไม่ดำเนินการต่อ

ส่วนที่ 4 : การพิมพ์ตั๋ว

Full Source From: `ticket_print_gui.py`

โค้ด `ticket_print_gui.py` มีหน้าที่หลักในการ แสดงรายละเอียดตั๋ว และสร้างไฟล์ PDF สำหรับการพิมพ์ตั๋ว โดยมีการติดต่อกับฐานข้อมูล SQL Server เพียง 1 จุด คือ การดึงข้อมูลพนักงานที่เข้าสู่ระบบ

```
def open_ticket_print_screen(prev_root, staff_id, bus_no, price):
    """สร้างหน้าจอ Ticket Print และผูกกับฟังก์ชันสร้าง PDF"""
    if prev_root:
        prev_root.destroy()

    print_root = tk.Tk()
    print_root.title("Ticket Print")
    print_root.geometry("300x500")
    print_root.config(bg="#F8E9E4")

    # 1. ดึงข้อมูลและกำหนดรายละเอียดตั๋ว
    user_data = get_logged_in_user_info(staff_id)
    staff_name = user_data[0] if user_data else "Staff ID N/A"
```

ในส่วนของข้อมูลอื่นๆ บนหน้าตั๋วนั้น ถูกส่งต่อโดยการเก็บค่าจากหน้าต่างๆก่อนมาถึงหน้านี้ หรือสรุปก็คือ หน้านี้มีหน้าที่แค่ปริ้นท์ตั๋ว และ ค่าถูกบันทึกลงใน DB ตั้งแต่หน้า ticket ticket_price_gui.py