ระบบจัดการข้อมูลนักศึกษา

Student information management system

นายศุภกร สิริเกื้อกูลชัย รหัส 6706022510174

นายณพนันท์ ศรีเกื้อกลิ่น รหัส 6706022510204

นายกิตติสินธุ์ วรรณวณิชภักดี รหัส 6706022510247

นางสาววริศรา ท้าวแก่ รหัส 6706022510166

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมสาสตรบัณฑิต

สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ คณะเทคโนโลยีและการจัดการอุตสาหกรรม

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ปีการศึกษา 2567

คำนำ

การจัดทำโครงงาน “ระบบจัดการข้อมูลนักศึกษา” นี้เป็นส่วนหนึ่งของวิชา COMPUTER PROGRAMMIMG ของหลักสูตรวิศวกรรมศาสตรบัณฑิตสาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศคณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ เพื่อให้นักศึกษาได้นำความรู้ที่เรียนมาทั้งหมดมาประยุกต์ใช้ในการพัฒนาโปรแกรมที่สามารถทำงานได้จริง โดยเน้นการออกแบบและเขียนโปรแกรมในภาษา Python ซึ่งเป็นภาษาที่เรียนมาในวิชา COMPUTER PROGRAMMING โดยโครงงานนี้จะช่วยการคิดวิเคราะห์และแก้ปัญหาทางเทคนิค เพื่อเตรียมความพร้อมในการประกอบอาชีพด้านวิศวกรรมสารสนเทศและเครือข่ายในอนาคต หากมีข้อผิดพลาดประการใด คณะผู้จัดทำต้องขออภัยไว้ ณ ที่นี้ด้วย

สารบัญ

สารบัญภาพ

ภาพที่ หน้า

สารบัญภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 วัตถุประสงค์ของดครงการ

1.1.1 เพื่อพัฒนาระบบที่สามารถจัดการระบบนักศึกษาได้อย่างมีประสิทธิภาพ

1.1.2 เพื่อฝึกทักษะการเขียนโปรแกรม Python

1.1.3 เพื่อฝึกกระบวนการคิดอย่างเป็นระบบ

1.1.4 เพื่อเรียนรู้การจัดการข้อมูลและไฟล์

1.2 ขอบเขตของโครงการ

1.2.1 ระบบการจัดการระบบนักศึกษา มีฟังก์ชั่นในการทำงาน 7 ฟังก์ชั่น ได้แก่

1.2.1.1 Add Student and Teacher : เพิ่มนักศึกษา และอาจารย์

Show Student and Teacher : แสดงข้อมูลนักศึกษา และอาจารย์

1.2.1.2 Get specific data: การค้นหาข้อมูลที่เฉพาะเจาะจง

1.2.1.3 Update Student and Teacher: อัพเดทข้อมูลนักศึกษา และอาจารย์

1.2.1.4 Delete data : ลบข้อมูลนักศึกษา และอาจารย์

1.2.1.5 Create Report: การสร้างรายงานเพื่อแสดงผลข้อมูล

1.2.1.6 Exit: ออกจาการทำงานของโปรแกรม

1.2.2 ฟิลด์ในการจัดทำระบบจัดการระบบนักศึกษา

1.2.2.7 Get student by id : รหัสนักศึกษา

1.2.2.8 Get student by name : ชื่อนักศึกษา

1.2.2.9 Get student by year : ชั้นปีของนักศึกษา

1.2.2.10 Get student by department : แผนกของนักศึกษา

1.2.2.11 Get teacher by id : รหัสแทนอาจารย์

1.2.2.12 Get teacher by name : ชื่ออาจารย์

1.3 วัตถุประสงค์ของโครงการ

1.3.1 พัฒนาระบบที่สามารถจัดการระบบนักศึกษาได้อย่างมีประสิทธิภาพ

1.3.2 ฝึกทักษะการเขียนโปรแกรม Python

1.3.3 ฝึกกระบวนการคิดอย่างเป็นระบบ

1.3.4 เรียนรู้การจัดการข้อมูลและไฟล์

1.4 เครื่องมือที่คาดว่าจะต้องใช้

1.4.1 ภาษาโปรแกรมที่ใช้ในการพัฒนาระบบ คือ Python สำหรับการพัฒนาโปรแกรมพื้นฐานที่เข้าใจง่ายและมีไลบรารีหลากหลายที่ช่วยจัดการข้อมูลได้ดี

1.4.2 Visual Studio Code เป็นโปรแกรม Text Editor ที่ออกแบบมาเพื่อให้เป็นเครื่องมือที่ช่วยในการเขียนโค้ด โดยสามารถรองรับภาษาโปรแกรมต่าง ๆ เช่น Python, JavaScript และอื่น ๆ

1.4.3 GitHub แพลตฟอร์มที่ช่วยในการเก็บโค้ดและทำงานร่วมกับผู้อื่นในการ

บทที่ 2
ระบบจัดการข้อมูลนักศึกษา

2.1 ฟิลด์ในระบบการจัดดารข้อมูลนักศึกษา

การจัดการข้อมูลหนังสือในระบบประกอบไปด้วย 5 ฟิลด์หลัก ซึ่งแต่ละฟิลด์มีรายละเอียดและความสำคัญ ดังนี้

2.1.1 Get student by id : เป็นฟิลด์สำหรับเก็บรหัสนักเรียน (Student ID) ซึ่งเป็นตัวเลขเฉพาะสำหรับนักเรียนแต่ละคน เช่น "123456", "142536", "415263" เป็นต้น

2.1.2 Get student by name : เป็นฟิลด์ที่ใช้เก็บชื่อของนักเรียน เช่น "Fluke", "Pee", "Non" เป็นต้น

2.1.3 Get student by year : ฟิลด์นี้ระบุจำนวนปีที่นักเรียนศึกษาในแผนกนั้น ๆ เช่น "1", "2" หมายถึงปีที่นักเรียนกำลังเรียนอยู่

2.1.4 Get student by department : ฟิลด์นี้ระบุแผนกหรือสาขาที่นักเรียนสังกัดอยู่ เช่น "INET" หรือ "INE" ซึ่งอาจเป็นชื่อย่อของแผนกการศึกษา

2.1.5 Get teacher by id : เป็นฟิลด์ที่เก็บรหัสประจำตัวของอาจารย์แต่ละคน ซึ่งอาจถูกใช้เพื่ออ้างอิงถึงอาจารย์เฉพาะบุคคล เช่น "Anirach" มีรหัส 2 "Chanathip" มีรหัส 1

2.1.6 Get teacher by name : ฟิลด์นี้เก็บชื่ออาจารย์ที่เป็นผู้ดูแลนักเรียนหรือสอนนักเรียนในรายวิชานั้น ๆ เช่น "Anirach", "Chanathip"

2.2 โครงสร้างและการทำงานของโปรแกรมระบบจัดการข้อมูลนักศึกษา

ระบบจัดการข้อมูลนักศึกษาออกแบบให้ทำงานผ่านเมนูหลัก ซึ่งเป็นจุดศูนย์กลางที่ควบคุมการทำงานทั้งหมดของโปรแกรม โดยเมนูหลักนี้อยู่ในไฟล์ main.py ที่ผู้ใช้ต้องเปิดเพื่อเริ่มใช้งานทุกครั้ง เมื่อเปิดไฟล์นี้ โปรแกรมจะโชว์เมนูให้ผู้ใช้เลือกทำงานตามฟังก์ชันต่างๆ เช่น เพิ่มข้อมูลนักเรียนหรือครู ดูข้อมูลที่มีอยู่ อัปเดตข้อมูล หรือลบข้อมูล รวมไปถึงการสร้างรายงานได้ในที่เดียว
ทุกอย่างในโปรแกรมจะถูกควบคุมผ่านไฟล์ main.py โดยที่ผู้ใช้ไม่ต้องเข้าไปยุ่งเกี่ยวกับโค้ดหรือไฟล์อื่นๆ ระบบออกแบบมาให้ใช้งานง่ายและสะดวกสบาย ทุกขั้นตอน ไม่ว่าจะเป็นการเพิ่ม ลบ หรืออัปเดตข้อมูล จะถูกจัดการและเก็บไว้ในระบบโดยอัตโนมัติ ทำให้ผู้ใช้สามารถทำงานทั้งหมดได้จากเมนูหลักเพียงอย่างเดียว ช่วยให้การจัดการข้อมูลเป็นไปอย่างราบรื่นและมีประสิทธิภาพ

ผู้ใช้สามารถเลือกตัวเลือกตามหมายเลขที่ต้องการ

กด 1 เพื่อเพิ่มข้อมูลนักเรียนหรือครู

กด 2 เพื่อดูข้อมูลที่มีอยู่

กด 3 เพื่อค้นหาข้อมูลเฉพาะ

กด 4 เพื่ออัปเดตข้อมูล

กด 5 เพื่อลบข้อมูล

กด 6 เพื่อสร้างรายงาน

กด 7 เพื่อออกจากโปรแกรม

การเรียกใช้ฟังก์ชันต่างๆ ตามการเลือกของผู้ใช้

เมื่อผู้ใช้ใส่หมายเลขเลือกแล้ว โปรแกรมจะตรวจสอบค่า และเรียกใช้ฟังก์ชันที่สอดคล้องตามการเลือกนั้น หากผู้ใช้เลือกหมายเลขที่ไม่ถูกต้อง จะมีการแจ้งเตือนให้ผู้ใช้ทราบและขอให้ใส่หมายเลขใหม่

การทำงานต่อเนื่อง

หลังจากการทำงานของฟังก์ชันที่ถูกเลือกเสร็จสิ้น ฟังก์ชัน main() จะถูกเรียกซ้ำเพื่อแสดงเมนูอีกครั้ง ทำให้โปรแกรมทำงานต่อเนื่อง จนกว่าผู้ใช้จะเลือกออกจากโปรแกรม

2.3 อธิบายการใช้งานตัวเลือกในเมนู

2.3.1 เพิ่มข้อมูล

- กดตัวเลข 1 เพื่อเลือกการเพิ่มข้อมูล
- ระบบจะถามคุณว่าจะเพิ่มข้อมูลนักเรียนหรือครู โดยให้เลือกตัวเลือกที่เหมาะสม
- หากเลือกนักเรียน:
  - ระบบจะขอให้คุณป้อนรหัสนักเรียน จากนั้นกด Enter
  - ถัดไป ระบบจะขอให้คุณป้อนชื่อของนักเรียน จากนั้นกด Enter
  - จากนั้น ระบบอาจจะถามข้อมูลเพิ่มเติม เช่น หมวดหมู่หรือสาขาวิชาที่เรียน ถ้ามีให้กรอกแล้วกด Enter
- หากเลือกครู:
  - ระบบจะขอให้คุณป้อนรหัสครู จากนั้นกด Enter
  - ถัดไป ระบบจะขอให้คุณป้อนชื่อของครู จากนั้นกด Enter

- เมื่อลงข้อมูลเสร็จ ระบบจะแจ้งเตือนว่า "เพิ่มข้อมูลเสร็จสิ้น" และกลับไปที่เมนูหลัก

2.3.2 แสดงข้อมูล

- กดตัวเลข 2 เพื่อเลือกการแสดงข้อมูล
- ระบบจะแสดงรายชื่อของนักเรียนหรือครูทั้งหมดในระบบ
- คุณสามารถเลื่อนดูข้อมูลได้ในหน้าจอหรือระบบจะแจ้งจำนวนข้อมูลที่มีอยู่
- หลังจากดูข้อมูลเสร็จ ระบบจะนำคุณกลับไปที่เมนูหลัก

2.3.3 ดึงข้อมูลเฉพาะ

- กดตัวเลข 3 เพื่อเลือกการดึงข้อมูลเฉพาะ
- ระบบจะถามให้คุณเลือกว่าจะดึงข้อมูลนักเรียนหรือครู
- หากเลือกนักเรียน:
    - ระบบจะขอให้คุณป้อนรหัสนักเรียนที่ต้องการค้นหา จากนั้นกด Enter
    - ระบบจะแสดงข้อมูลนักเรียนที่ตรงกับรหัสที่ป้อน ถ้ามีข้อมูล
    - หากไม่พบข้อมูล ระบบจะแจ้งว่า "ไม่พบข้อมูลนักเรียน"
- หากเลือกครู:
    - ระบบจะขอให้คุณป้อนรหัสครูที่ต้องการค้นหา จากนั้นกด Enter
    - ระบบจะแสดงข้อมูลครูที่ตรงกับรหัสที่ป้อน ถ้ามีข้อมูล
    - หากไม่พบข้อมูล ระบบจะแจ้งว่า "ไม่พบข้อมูลครู"
- เมื่อดึงข้อมูลเสร็จ ระบบจะนำคุณกลับไปที่เมนูหลัก

2.3.4 อัปเดตข้อมูล

- กดตัวเลข 4 เพื่อเลือกการอัปเดตข้อมูล
- ระบบจะถามให้คุณเลือกว่าจะอัปเดตข้อมูลนักเรียนหรือครู
- หากเลือกนักเรียน:
    - ระบบจะขอให้คุณป้อนรหัสนักเรียนที่ต้องการอัปเดต จากนั้นกด Enter
    - ระบบจะแสดงข้อมูลปัจจุบันของนักเรียนที่เลือก
    - ระบบจะขอให้คุณป้อนข้อมูลใหม่ เช่น ชื่อหรือข้อมูลอื่น ๆ โดยสามารถแก้ไขข้อมูลตามต้องการ
    - หลังจากกรอกข้อมูลใหม่เสร็จ ระบบจะยืนยันการอัปเดตข้อมูล
- หากเลือกครู:

ระบบจะขอให้คุณป้อนรหัสครูที่ต้องการอัปเดต จากนั้นกด Enter

ระบบจะแสดงข้อมูลปัจจุบันของครูที่เลือก

ระบบจะขอให้คุณป้อนข้อมูลใหม่ เช่น ชื่อหรือข้อมูลอื่น ๆ โดยสามารถแก้ไขข้อมูลตามต้องการ

หลังจากกรอกข้อมูลใหม่เสร็จ ระบบจะยืนยันการอัปเดตข้อมูล

เมื่อทำการอัปเดตเสร็จ ระบบจะแจ้งว่า "อัปเดตข้อมูลเสร็จสิ้น" และกลับไปที่เมนูหลัก

2.3.5 ลบข้อมูล

กดตัวเลข 5 เพื่อเลือกการลบข้อมูล

ระบบจะแจ้งให้คุณป้อนรหัสนักเรียนหรือรหัสครูที่ต้องการลบ

ป้อนรหัสแล้วกด Enter

ระบบจะตรวจสอบว่ารหัสที่กรอกมีอยู่ในระบบหรือไม่

หากรหัสที่กรอกไม่มีในระบบ ระบบจะแจ้งเตือนว่า "ไม่พบข้อมูล" และให้คุณกลับไปที่เมนูหลัก

หากรหัสถูกต้อง ระบบจะทำการค้นหาข้อมูลที่ตรงกันและแสดงรายละเอียดของนักเรียนหรือครูที่ต้องการลบ เช่น รหัส ชื่อ และข้อมูลอื่น ๆ ที่เกี่ยวข้อง เพื่อให้คุณสามารถตรวจสอบได้ว่าต้องการลบข้อมูลนี้หรือไม่

ระบบจะแจ้งเตือนเพื่อยืนยันการลบข้อมูล โดยอาจมีข้อความถามว่า "คุณแน่ใจหรือไม่ว่าจะลบข้อมูลนี้?" คุณจะต้องกดยืนยันโดยการพิมพ์ Y (ใช่) เพื่อดำเนินการลบ หรือกด N (ไม่ใช่) เพื่อยกเลิกการลบ

หากคุณกด Y ระบบจะทำการลบข้อมูลนั้นจากฐานข้อมูล

หากคุณกด N ระบบจะกลับไปที่เมนูหลักโดยไม่ทำการลบข้อมูลใด ๆ

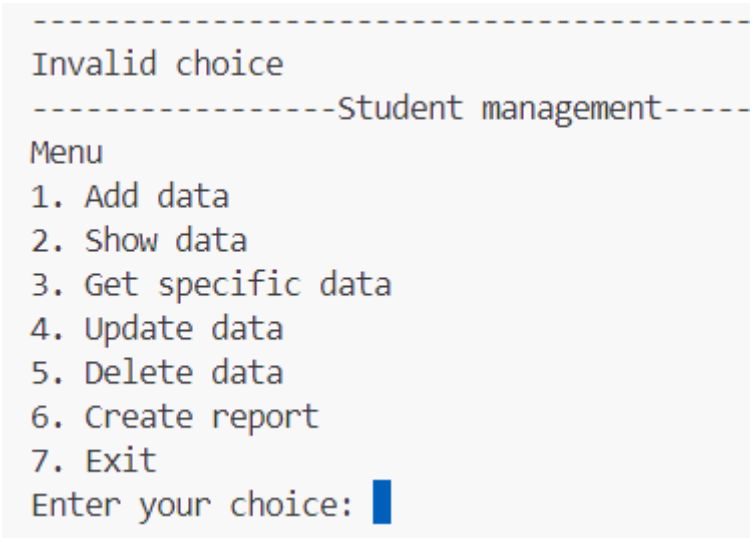
เมื่อทำการลบข้อมูลเสร็จสิ้น ระบบจะแสดงข้อความยืนยันว่า "การลบข้อมูลเสร็จสิ้น" และจะนำคุณกลับไปที่เมนูหลัก

2.3.6 สร้างรายงาน

กดตัวเลข 6 เพื่อเลือกการสร้างรายงาน

2.3.7 ออกจากโปรแกรม

กดตัวเลข 7 เพื่อเลือกออกจากโปรแกรม



ภาพที่ 2-1 ตัวอย่างหน้าโปรแกรมหลัก

บทที่ 3
โครงสร้างและการทำงานของแต่ละไฟล์

ในระบบการจัดการข้อมูลทางการศึกษา โครงสร้างโปรแกรมที่ดีมีความสำคัญต่อการจัดการข้อมูลนักเรียนและครูอย่างมีประสิทธิภาพและสะดวกต่อผู้ใช้ ในการอธิบายโค้ดแต่ละไฟล์ เราจะเริ่มต้นจากไฟล์ main.py ซึ่งเป็นจุดเริ่มต้นของโปรแกรมที่เชื่อมต่อฟังก์ชันการทำงานทั้งหมดเข้าด้วยกัน ไฟล์นี้มีหน้าที่หลักในการเรียกใช้งานฟังก์ชันต่าง ๆ เพื่อให้ผู้ใช้สามารถทำงานกับข้อมูลได้อย่างราบรื่น

ถัดไปคือไฟล์ menu.py ซึ่งมีคลาส MenuClass ที่ช่วยในการสร้างเมนูและการโต้ตอบกับผู้ใช้ โดยเมนูนี้มีความสำคัญในการให้ผู้ใช้เลือกฟังก์ชันต่าง ๆ ที่ต้องการใช้งานอย่างชัดเจนและเข้าใจง่าย การออกแบบเมนูที่ดีช่วยให้ผู้ใช้สามารถนำทางในโปรแกรมได้อย่างสะดวก

ต่อมาคือไฟล์ report.py ซึ่งมีคลาส ReportClass ที่รับผิดชอบในการจัดการและสร้างรายงานข้อมูลที่สำคัญของนักเรียนและครูในระบบ คลาสนี้ช่วยให้ผู้ใช้สามารถเข้าถึงข้อมูลในรูปแบบที่เข้าใจง่ายและเหมาะสมสำหรับการวิเคราะห์หรือการจัดการข้อมูลทางการศึกษา

จากนั้นเราจะพูดถึงไฟล์ student.py ซึ่งมีการออกแบบคลาส StudentClass ที่จัดการข้อมูลนักเรียน รวมถึงการเพิ่ม แก้ไข ลบ และดึงข้อมูลนักเรียน การใช้ไฟล์นี้ช่วยให้การจัดการข้อมูลนักเรียนทำได้อย่างมีประสิทธิภาพและแม่นยำ

สุดท้ายคือไฟล์ teacher.py ที่มีคลาส TeacherClass ซึ่งทำหน้าที่คล้ายกับคลาส StudentClass แต่สำหรับข้อมูลของครู การมีโครงสร้างที่ชัดเจนในไฟล์นี้ช่วยให้สามารถจัดการข้อมูลครูได้อย่างมีประสิทธิภาพและถูกต้อง

การออกแบบโปรแกรมในลักษณะนี้ช่วยให้ผู้ใช้สามารถเข้าถึงข้อมูลได้อย่างมีระบบและง่ายดาย โดยการจัดการข้อมูลจะเป็นไปอย่างราบรื่นและมีประสิทธิภาพในทุกขั้นตอน

3.1 ไฟล์: main.py

ไฟล์ main.py เป็นจุดเริ่มต้นของโปรแกรมที่ออกแบบมาเพื่อจัดการข้อมูลนักเรียนและครูในระบบการศึกษา โดยมีวัตถุประสงค์เพื่อให้ผู้ใช้สามารถทำการเพิ่มข้อมูล อัปเดตข้อมูล ลบข้อมูล และดึงข้อมูลต่าง ๆ จากไฟล์ที่ถูกเก็บในรูปแบบไบต์ ซึ่งเป็นการจัดเก็บข้อมูลในไฟล์ .bin ที่ไม่สามารถอ่านได้โดยตรงด้วยสายตาของมนุษย์ แต่เหมาะสมสำหรับการทำงานภายในโปรแกรมที่ใช้จัดการข้อมูลขนาดใหญ่และมีโครงสร้างที่แน่นอน

การทำงานหลักของไฟล์ main.py

โปรแกรมนี้ออกแบบมาให้สามารถรองรับการทำงานหลายอย่างที่เกี่ยวข้องกับการจัดการข้อมูลของนักเรียนและครู โดยการทำงานหลัก ๆ จะครอบคลุมทั้งการเพิ่มข้อมูลใหม่ การแสดงข้อมูลทั้งหมด การค้นหาข้อมูลเฉพาะเจาะจง การอัปเดตข้อมูลที่มีอยู่ และการลบข้อมูลที่ไม่ต้องการ นอกจากนี้ยังมีการสร้างรายงานจากข้อมูลที่เก็บไว้เพื่อการวิเคราะห์หรือการนำไปใช้งานต่อไป

การจัดการข้อมูลแบบไบต์

การที่ข้อมูลถูกจัดเก็บในรูปแบบไบต์ (.bin) ทำให้การจัดการข้อมูลในโปรแกรมนี้มีประสิทธิภาพมากขึ้นเมื่อเปรียบเทียบกับการจัดเก็บในรูปแบบไฟล์ที่มนุษย์อ่านได้ เช่น ไฟล์ข้อความ (.txt) หรือไฟล์ CSV ไฟล์แบบไบต์ช่วยให้โปรแกรมสามารถบันทึกและดึงข้อมูลได้เร็วขึ้น และลดปัญหาความซับซ้อนในการประมวลผลข้อมูลขนาดใหญ่ โดยเฉพาะอย่างยิ่งในการจัดเก็บข้อมูลหลายฟิลด์ของนักเรียนและครู เช่น หมายเลขประจำตัว ชื่อ ปีการศึกษา หรือแผนกการศึกษา

ฟังก์ชันการทำงานที่ครอบคลุม

โปรแกรมในไฟล์ main.py จะทำการเรียกใช้ฟังก์ชันจากคลาสต่าง ๆ ที่ถูกออกแบบในไฟล์อื่น ๆ เช่น คลาส StudentClass สำหรับการจัดการข้อมูลนักเรียน และคลาส TeacherClass สำหรับการจัดการข้อมูลครู ซึ่งทั้งสองคลาสนี้มีฟังก์ชันการทำงานที่ครอบคลุมความต้องการในการจัดการข้อมูลในระบบอย่างครบถ้วน เช่น ฟังก์ชัน add_student สำหรับเพิ่มข้อมูลนักเรียนใหม่ ฟังก์ชัน update_student สำหรับอัปเดตข้อมูลนักเรียนที่มีอยู่ หรือฟังก์ชัน remove_teacher ที่ใช้ในการลบข้อมูลครูออกจากระบบ

ในส่วนของการแสดงผลข้อมูล โปรแกรมนี้ออกแบบให้สามารถแสดงข้อมูลนักเรียนหรือครูทั้งหมดที่ถูกจัดเก็บไว้ในไฟล์ โดยฟังก์ชัน get_all_student และ get_all_teacher จะช่วยให้ผู้ใช้สามารถดึง

ข้อมูลนักเรียนและครูทั้งหมดออกมาแสดงได้ในรูปแบบที่อ่านง่ายและจัดเรียงตามฟิลด์ต่าง ๆ ที่ต้องการ

ความยืดหยุ่นในการดึงข้อมูลเฉพาะเจาะจง

โปรแกรมนี้ยังมีความยืดหยุ่นในการดึงข้อมูลเฉพาะเจาะจงตามเงื่อนไขที่ผู้ใช้กำหนด เช่น ผู้ใช้สามารถค้นหานักเรียนโดยใช้หมายเลขประจำตัวนักเรียนหรือชื่อของนักเรียนได้โดยตรง โดยไม่จำเป็นต้องเลื่อนดูข้อมูลทั้งหมด ฟังก์ชัน get_student_by_id และ get_student_by_name จะทำการค้นหานักเรียนที่มีหมายเลขประจำตัวหรือชื่อตรงกับเงื่อนไขที่ผู้ใช้ป้อนเข้ามา ฟังก์ชันนี้จะช่วยประหยัดเวลาและทำให้การจัดการข้อมูลเป็นไปอย่างมีประสิทธิภาพ

นอกจากการค้นหานักเรียนตามหมายเลขประจำตัวหรือชื่อแล้ว โปรแกรมนี้ยังรองรับการค้นหาข้อมูลนักเรียนตามปีการศึกษาและแผนกการศึกษา ฟังก์ชัน get_student_by_year และ get_student_by_department จะดึงข้อมูลของนักเรียนที่อยู่ในปีการศึกษาและแผนกที่ผู้ใช้ต้องการ ทำให้สามารถดูข้อมูลตามเงื่อนไขเฉพาะได้อย่างรวดเร็ว นอกจากนี้ยังสามารถค้นหานักเรียนตามหมายเลขประจำตัวของครูที่ดูแลนักเรียนคนนั้น ๆ ได้ผ่านฟังก์ชัน get_student_by_teacher_id ซึ่งเป็นฟังก์ชันที่ออกแบบมาเพื่อรองรับการจัดการข้อมูลในระบบการศึกษาโดยเฉพาะ

การจัดการข้อมูลครู

ในส่วนของข้อมูลครู โปรแกรมนี้มีฟังก์ชันที่ออกแบบมาให้รองรับการจัดการข้อมูลของครูในลักษณะเดียวกันกับนักเรียน ผู้ใช้สามารถเพิ่มข้อมูลครูใหม่โดยใช้ฟังก์ชัน add_teacher ซึ่งทำงานคล้ายกับฟังก์ชัน add_student โดยจะมีการตรวจสอบความถูกต้องของข้อมูลก่อนที่จะบันทึกลงในไฟล์ นอกจากนี้ยังสามารถค้นหาข้อมูลครูตามหมายเลขประจำตัวหรือชื่อของครูได้เช่นเดียวกับการค้นหานักเรียน โปรแกรมยังรองรับการลบข้อมูลครูที่ไม่ต้องการออกจากระบบด้วยฟังก์ชัน remove_teacher

การสร้างรายงาน

โปรแกรมในไฟล์ main.py ยังสามารถสร้างรายงานจากข้อมูลที่มีอยู่ในระบบได้ โดยผู้ใช้สามารถดึงข้อมูลนักเรียนหรือครูทั้งหมดแล้วนำไปใช้ในการวิเคราะห์หรือสร้างเอกสารรายงานเพิ่มเติม การสร้างรายงานช่วยให้ผู้ใช้สามารถสรุปผลข้อมูลและนำเสนอได้อย่างเป็นระบบและชัดเจน

การออกแบบเพื่อความยืดหยุ่นและการขยายในอนาคต

ไฟล์ main.py นี้ถูกออกแบบมาให้สามารถขยายการทำงานในอนาคตได้อย่างง่ายดาย ผู้พัฒนาสามารถเพิ่มฟังก์ชันใหม่ ๆ เพื่อรองรับความต้องการเพิ่มเติมโดยไม่จำเป็นต้องแก้ไขโครงสร้างหลักของโปรแกรม เช่น การเพิ่มระบบการจัดการข้อมูลผลการเรียนของนักเรียนหรือการจัดการข้อมูลคลาสเรียนต่าง ๆ ซึ่งจะทำให้โปรแกรมนี้มีความสามารถที่หลากหลายขึ้นและรองรับการใช้งานในสภาพแวดล้อมที่เปลี่ยนแปลงไป

main.py เป็นจุดเริ่มต้นที่สำคัญของโปรแกรมจัดการข้อมูลนักเรียนและครู โดยมีฟังก์ชันการทำงานที่ครบถ้วนสำหรับการจัดการข้อมูลอย่างมีประสิทธิภาพ พร้อมด้วยความยืดหยุ่นในการค้นหาและดึงข้อมูลตามเงื่อนไขเฉพาะ และยังสามารถขยายการทำงานในอนาคตเพื่อรองรับความต้องการเพิ่มเติม

3.1.1 นำเข้า MenuClass จากไฟล์ menu.py

ซึ่งเป็นคลาสที่มีฟังก์ชันต่าง ๆ สำหรับจัดการเมนูของโปรแกรม เช่น การเพิ่มข้อมูล การแสดงข้อมูล ฯลฯ นอกจากนี้เรายังนำเข้าโมดูล os ซึ่งจะใช้ในการตรวจสอบการมีอยู่ของไฟล์บนระบบปฏิบัติการ เช่น เช็กว่าไฟล์ข้อมูลนักเรียนและครูมีอยู่แล้วหรือไม่

```
from menu import MenuClass
import os
```

ภาพที่ 3-1 การนำเข้าคลาส

3.1.2 การสร้างฟังก์ชัน main

ที่นี่เราสร้างฟังก์ชัน main ซึ่งเป็นฟังก์ชันหลักที่โปรแกรมจะทำงานเมื่อเริ่มต้น เมื่อเข้าไปในฟังก์ชันนี้ เราจะสร้างออบเจ็กต์ menu จากคลาส MenuClass ซึ่งจะทำให้เราสามารถเรียกใช้ฟังก์ชันต่าง ๆ ที่มีอยู่ในคลาสนี้ได้

```
def main():
    menu = MenuClass()
```

ภาพที่ 3-2 สร้างฟังก์ชัน main

3.1.3 เช็กไฟล์ที่ใช้เก็บข้อมูลมีอยู่ในระบบหรือไม่

โปรแกรมจะเช็กว่าไฟล์ student.bin และ teacher.bin มีอยู่ในระบบหรือไม่ หากไฟล์เหล่านี้ไม่มีอยู่ โปรแกรมจะแจ้งข้อความว่า "Creating student.bin..." หรือ "Creating teacher.bin..." เพื่อให้ผู้ใช้ทราบว่ากำลังสร้างไฟล์ใหม่ และจะใช้คำสั่ง with open(...) เพื่อสร้างไฟล์ใหม่ในโหมด wb (write binary) โดยที่ไม่มีข้อมูลใด ๆ ถูกเขียนลงไปในไฟล์ในขณะนี้

```
if not os.path.exists('student.bin'):
    print("Creating student.bin...")
    with open('student.bin', 'wb') as file:
        pass
if not os.path.exists('teacher.bin'):
    print("Creating teacher.bin...")
    with open('teacher.bin', 'wb') as file:
        pass
```

ภาพที่ 3-3 เช็คและสร้างไฟล์

3.1.4 แสดงเมนูการจัดการข้อมูล

ที่นี่เรามีลูป while True ซึ่งทำให้โปรแกรมแสดงเมนูการจัดการข้อมูลอย่างต่อเนื่อง เมนูนี้จะมีตัวเลือกให้ผู้ใช้เลือกทำงานต่าง ๆ ได้แก่ การเพิ่มข้อมูล (Add data) การแสดงข้อมูล (Show data) การรับข้อมูลเฉพาะ (Get specific data) การอัปเดตข้อมูล (Update data) การลบข้อมูล (Delete data) การสร้างรายงาน (Create report) และการออกจากโปรแกรม (Exit)

```
while True:
    print('-----------------Student management-----------------')
    print('Menu')
    print('1. Add data')
    print('2. Show data')
    print('3. Get specific data')
    print('4. Update data')
    print('5. Delete data')
    print('6. Create report')
    print('7. Exit')
```

ภาพที่ 3-4 แสดงเมนูการจัดการข้อมูล

3.1.5 ตรวจข้อมูลที่ผู้ใช้กรอก

โปรแกรมจะรอให้ผู้ใช้กรอกตัวเลือกจากเมนู เมื่อผู้ใช้กรอกข้อมูลเสร็จ โปรแกรมจะแสดงเส้นแบ่งเพื่อให้เห็นภาพชัดเจนขึ้น หากเกิดข้อผิดพลาดในการกรอกข้อมูล โปรแกรมจะจับข้อผิดพลาดด้วยคำสั่ง except และแสดงข้อความ "Invalid choice" และกลับไปเริ่มใหม่ในลูป

```
try:
    choice = input('Enter your choice: ')
    print('-----------------------------------------------------')
except:
    print('Invalid choice')
    continue
```

ภาพที่ 3-5 ตรวจข้อมูลที่ผู้ใช้กรอก

3.1.6 ตรวจสอบตัวเลือกที่ผู้ใช้เลือก

ในส่วนนี้ โปรแกรมจะตรวจสอบตัวเลือกที่ผู้ใช้เลือก หากผู้ใช้เลือกตัวเลือกที่ถูกต้อง เช่น หากเลือก "1" จะเรียกใช้งานฟังก์ชัน add_data() จาก menu เพื่อเพิ่มข้อมูล ถ้าเลือก "2" จะเรียกใช้ show_data() เพื่อแสดงข้อมูล และทำเช่นนี้กับตัวเลือกอื่น ๆ จนถึงตัวเลือก "7" ซึ่งจะทำให้โปรแกรมหยุดทำงาน (break)

```
if choice == '1':
    menu.add_data()
elif choice == '2':
    menu.show_data()
elif choice == '3':
    menu.get_specific_data()
elif choice == '4':
    menu.update_data()
elif choice == '5':
    menu.delete_data()
elif choice == '6':
    menu.create_report()
elif choice == '7':
    break
else:
    print('Invalid choice')
```

ภาพที่ 3-6 ตรวจสอบตัวเลือกที่ผู้ใช้เลือก

3.1.7 เรียกใช้ฟังก์ชัน main

เราเรียกใช้ฟังก์ชัน main เพื่อเริ่มต้นการทำงานของโปรแกรมทั้งหมด เมื่อโปรแกรมทำงาน ฟังก์ชันนี้จะทำให้โปรแกรมเริ่มทำงานตามลำดับที่เราได้กำหนดไว้

```
main()
```

ภาพที่ 3-7 เรียกใช้ฟังก์ชัน main

โปรแกรมในไฟล์ main.py เป็นโครงสร้างหลักของระบบจัดการข้อมูลนักเรียนและครู โดยมีฟังก์ชันการทำงานที่ชัดเจนในการเช็กไฟล์ ขึ้นเมนูให้ผู้ใช้เลือกทำงานต่าง ๆ และจัดการข้อมูลในไฟล์แบบไบต์ โปรแกรมนี้สามารถใช้ในการจัดการข้อมูลอย่างมีประสิทธิภาพและง่ายต่อการเข้าใจ

3.2 ไฟล์ menu.py

ไฟล์ menu.py เป็นส่วนสำคัญในโปรแกรมการจัดการข้อมูลการศึกษาที่ทำหน้าที่เป็นอินเทอร์เฟซระหว่างผู้ใช้และระบบ โดยมีการใช้คลาส MenuClass ในการจัดการคำสั่งต่าง ๆ ที่เกี่ยวข้องกับข้อมูลของนักเรียนและครูในระบบการศึกษา ฟังก์ชันภายใน MenuClass ได้รับการออกแบบมาเพื่อให้ผู้ใช้สามารถโต้ตอบกับระบบได้อย่างมีประสิทธิภาพและสะดวกสบาย รวมถึงมีความสามารถในการตรวจสอบและจัดการข้อผิดพลาดต่าง ๆ ที่อาจเกิดขึ้น

ภายในคลาส MenuClass มีการจัดการเมนูคำสั่งต่าง ๆ ที่ครอบคลุมการทำงานสำคัญ ๆ เช่น การเพิ่มนักเรียนและครูใหม่ การค้นหาข้อมูล การอัปเดต การลบข้อมูล และการแสดงรายงานข้อมูลทั้งหมด โดยทุกคำสั่งถูกจัดทำให้อยู่ในรูปแบบที่ผู้ใช้สามารถเลือกใช้ได้ง่าย นอกจากนี้ระบบยังมีการตรวจสอบความถูกต้องของข้อมูล เช่น การตรวจสอบว่าหมายเลขประจำตัวของนักเรียนหรือครูที่ป้อนเข้ามาไม่มีการซ้ำกัน หรือการตรวจสอบความถูกต้องของข้อมูลการศึกษา

เมนูการทำงานแต่ละอย่างถูกออกแบบให้ทำงานได้อย่างอัตโนมัติและมีการตอบสนองต่อข้อผิดพลาดอย่างเหมาะสม หากผู้ใช้ป้อนข้อมูลที่ไม่ถูกต้อง ระบบจะทำการแจ้งเตือนข้อผิดพลาดเพื่อให้แก้ไขและป้อนข้อมูลใหม่ นอกจากนี้ยังมีการจัดการสถานการณ์ที่ไม่มีข้อมูลในระบบ โดยจะแจ้งให้ผู้ใช้ทราบถึงการขาดข้อมูลหรือตัวเลือกที่ไม่สามารถดำเนินการได้ เช่น การค้นหานักเรียนที่ไม่มีอยู่ในระบบ หรือการพยายามลบข้อมูลที่ไม่มีอยู่จริง

อีกทั้งระบบนี้ยังสามารถปรับแต่งและขยายเพิ่มเติมได้ง่ายในอนาคต หากต้องการเพิ่มคำสั่งใหม่หรือปรับเปลี่ยนรูปแบบการจัดการข้อมูล เช่น การเพิ่มฟังก์ชันในการจัดการผลการเรียนหรือข้อมูลอื่น ๆ ของนักเรียนและครู ระบบโครงสร้างเมนูจะรองรับการปรับปรุงโดยไม่ทำให้การทำงานส่วนอื่น ๆ ของโปรแกรมได้รับผลกระทบ ทำให้โปรแกรมมีความยืดหยุ่นและพร้อมสำหรับการพัฒนาต่อในอนาคต

การออกแบบนี้ทำให้ menu.py กลายเป็นส่วนสำคัญที่ทำให้ระบบมีการทำงานที่ง่ายต่อการใช้งาน แต่ยังคงความครอบคลุมในการจัดการข้อมูลที่ซับซ้อน อีกทั้งการใช้ไฟล์ไบนารีในการเก็บข้อมูลยังช่วยเพิ่มความปลอดภัยและการทำงานที่รวดเร็ว

3.2.1 ฟังก์ชัน add_data

ซึ่งเริ่มต้นด้วยการให้ผู้ใช้เลือกประเภทของข้อมูลที่ต้องการเพิ่ม เมื่อผู้ใช้เลือกเพิ่มนักเรียน ระบบจะทำการตรวจสอบว่ามีครูในระบบหรือไม่ ถ้าไม่มี จะไม่สามารถเพิ่มนักเรียนได้ เนื่องจากข้อมูลนักเรียนจะต้องมีการเชื่อมโยงกับครูที่ดูแล

ในกรณีที่ผู้ใช้เลือกเพิ่มนักเรียน ระบบจะขอให้ผู้ใช้กรอกรหัสนักเรียน ชื่อ ปีการศึกษา แผนก และเลือกครูที่ดูแลนักเรียน หากมีการกรอกข้อมูลไม่ถูกต้อง ระบบจะแจ้งข้อผิดพลาดและไม่ทำการเพิ่มข้อมูล หากเลือกเพิ่มครู ผู้ใช้จะกรอกรหัสและชื่อของครูและสามารถเพิ่มข้อมูลได้ทันที

```
def add_data(self):
    # ... (code)
```

ภาพที่ 3-8 ฟังก์ชัน add_data

3.2.2 ฟังก์ชัน show_data

ฟังก์ชันนี้ใช้สำหรับแสดงข้อมูลนักเรียนและครูทั้งหมดในระบบ โดยจะแบ่งการแสดงข้อมูลออกเป็นสองส่วนหลัก: ข้อมูลนักเรียนและข้อมูลครู หากไม่มีข้อมูลในระบบ ฟังก์ชันจะแจ้งให้ผู้ใช้ทราบว่าข้อมูลนั้นไม่มีอยู่ โดยฟังก์ชันนี้เรียกใช้ข้อมูลจากคลาส StudentClass และ TeacherClass เพื่อดึงข้อมูลและทำการแสดงผลลัพธ์ที่ต้องการ

ในการแสดงข้อมูล นักเรียนแต่ละคนจะแสดงรหัส ชื่อ ปีการศึกษา แผนก และรหัสครูที่ดูแลอยู่ ขณะที่ข้อมูลครูจะแสดงรหัสและชื่อของครู หากไม่มีนักเรียนหรือครูในระบบ ฟังก์ชันจะแจ้งให้ทราบอย่างชัดเจน

```
def show_data(self):
    # ... (code)
```

ภาพที่ 3-9 ฟังก์ชัน show_data

3.2.3 ฟังก์ชัน get_specific_data

ฟังก์ชันนี้ให้ผู้ใช้สามารถค้นหาข้อมูลนักเรียนหรือครูตามเงื่อนไขที่กำหนด เช่น รหัสนักเรียน ชื่อ ปีการศึกษา หรือรหัสครู โดยเริ่มต้นด้วยการให้ผู้ใช้เลือกว่าจะค้นหานักเรียนหรือครู หลังจากนั้นจะแบ่งประเภทการค้นหาออกเป็นหลายแบบ เช่น ค้นหานักเรียนตามรหัส ชื่อ ปีการศึกษา แผนก หรือรหัสครูที่ดูแล

สำหรับการค้นหานักเรียน หากผู้ใช้เลือกค้นหาตามรหัส ระบบจะขอให้กรอกรหัสนักเรียนและทำการค้นหาข้อมูล หากพบจะทำการแสดงข้อมูลออกมาให้ผู้ใช้เห็น หากไม่พบข้อมูล ระบบจะแจ้งให้ทราบว่าข้อมูลนั้นไม่มีอยู่ ในกรณีที่เป็นการค้นหาครู ก็จะมีวิธีการค้นหาที่คล้ายคลึงกัน

```
def get_specific_data(self):
    # ... (code)
```

ภาพที่ 3-10 ฟังก์ชัน get_specific_data

3.2.4 ฟังก์ชัน update_data

ฟังก์ชันนี้ช่วยให้ผู้ใช้สามารถอัปเดตข้อมูลนักเรียนหรือครูที่มีอยู่ในระบบได้ โดยจะเริ่มต้นด้วยการให้ผู้ใช้เลือกประเภทของข้อมูลที่ต้องการอัปเดต หลังจากนั้นจะแสดงรายชื่อนักเรียนหรือครูทั้งหมดในระบบ เพื่อให้ผู้ใช้เลือกคนที่ต้องการอัปเดต โดยจะต้องกรอกข้อมูลใหม่สำหรับฟิลด์ที่ต้องการเปลี่ยนแปลง

ในกรณีที่ผู้ใช้ต้องการอัปเดตข้อมูลนักเรียน ระบบจะให้กรอกข้อมูลใหม่ เช่น ชื่อ ปีการศึกษา แผนก และรหัสครูที่ดูแล โดยผู้ใช้สามารถข้ามฟิลด์ที่ไม่ต้องการเปลี่ยนแปลงได้ เมื่อกรอกข้อมูลครบถ้วนแล้ว ระบบจะทำการอัปเดตข้อมูลในระบบ

```
def update_data(self):
    # ... (code)
```

ภาพที่ 3-11 ฟังก์ชัน update_data

3.2.5 ฟังก์ชัน delete_data

ฟังก์ชันนี้ใช้เพื่อลบข้อมูลนักเรียนหรือครูออกจากระบบ โดยเริ่มต้นด้วยการให้ผู้ใช้เลือกประเภทของข้อมูลที่ต้องการลบ ซึ่งหากผู้ใช้เลือกลบนักเรียน ระบบจะแสดงรายชื่อนักเรียนทั้งหมดให้เลือกและจะทำการลบข้อมูลตามที่เลือก หากผู้ใช้เลือกลบครู ระบบจะแจ้งเตือนว่าการลบครูจะมีผลทำให้นักเรียนที่อยู่ภายใต้การดูแลของครูนั้นถูกลบด้วยเช่นกัน

ระบบจะทำการลบข้อมูลของครูและนักเรียนที่เกี่ยวข้องออกจากระบบ หากผู้ใช้กรอกข้อมูลไม่ถูกต้อง ระบบจะแจ้งข้อผิดพลาดและไม่ทำการลบข้อมูล

```
def delete_data(self):
    # ... (code)
```

ภาพที่ 3-12 ฟังก์ชัน delete_data

3.2.6 ฟังก์ชัน create_report

ฟังก์ชันนี้จะทำการสร้างรายงานข้อมูลนักเรียนและครู โดยจะรวบรวมข้อมูลทั้งหมดที่มีในระบบและเขียนลงในไฟล์ชื่อ report.txt ในรูปแบบที่เข้าใจง่าย โดยข้อมูลที่บันทึกจะประกอบไปด้วยชื่อ รหัส ปีการศึกษา และแผนกของนักเรียน รวมถึงรหัสและชื่อของครู

รายงานนี้สามารถนำไปใช้ในการวิเคราะห์ข้อมูลหรือส่งให้ผู้ที่เกี่ยวข้องได้อย่างสะดวก โดยเมื่อเสร็จสิ้นการสร้างรายงาน ระบบจะแจ้งให้ผู้ใช้ทราบว่าได้ทำการสร้างรายงานเรียบร้อยแล้ว

```
def create_report(self):
    # ... (code)
```

ภาพที่ 3-13 ฟังก์ชัน create_report

ไฟล์ menu.py ถือเป็นส่วนสำคัญที่ช่วยจัดการข้อมูลนักเรียนและครูในระบบการศึกษา โดยจัดเตรียมฟังก์ชันที่หลากหลายให้ผู้ใช้สามารถทำการเพิ่ม แก้ไข ลบ แสดงข้อมูล และสร้างรายงานข้อมูลได้อย่างมีประสิทธิภาพ การออกแบบระบบให้มีการตรวจสอบความถูกต้องของข้อมูลและการจัดการข้อผิดพลาดที่ดี ทำให้ผู้ใช้สามารถใช้งานได้อย่างสะดวกและปลอดภัย ระบบนี้ไม่เพียงแต่ช่วยในการจัดการข้อมูล แต่ยังช่วยในการวางแผนและการประเมินผลในด้านการศึกษาได้อย่างมีประสิทธิภาพ

3.3 ไฟล์ report.py

ไฟล์ report.py ประกอบด้วยคลาส ReportClass ซึ่งมีบทบาทสำคัญในการจัดการและสร้างรายงานที่ครอบคลุมข้อมูลของนักเรียนและครูในระบบการศึกษา โดยคลาสนี้ถูกออกแบบมาเพื่ออำนวยความสะดวกให้ผู้ใช้สามารถเข้าถึงและสรุปข้อมูลสำคัญเกี่ยวกับสถานะของนักเรียนและครูอย่างเป็นระบบและเข้าใจง่าย ฟังก์ชันต่าง ๆ ที่มีอยู่ในคลาส ReportClass มีความสามารถในการรวบรวมข้อมูลที่ซับซ้อนและนำเสนอในรูปแบบที่เหมาะสมกับการวิเคราะห์และการตัดสินใจในการจัดการข้อมูลทางการศึกษา

การทำงานของคลาส ReportClass มีการจัดการข้อมูลจากไฟล์ไบนารี (.bin) ที่เก็บข้อมูลนักเรียนและครู ซึ่งเป็นรูปแบบที่เหมาะสมสำหรับการเก็บข้อมูลขนาดใหญ่และการอ่านเขียนข้อมูลอย่างมีประสิทธิภาพ แต่ข้อมูลในรูปแบบไบนารีนั้นไม่สามารถอ่านได้โดยตรง ดังนั้นจึงมีฟังก์ชันที่ช่วยแปลงข้อมูลเหล่านี้ให้อยู่ในรูปแบบที่มนุษย์สามารถอ่านได้ เช่น การสร้างรายงานสรุปนักเรียนทั้งหมดที่ลงทะเบียนในระบบ การแสดงรายชื่อนักเรียนตามปีการศึกษา หรือการจัดกลุ่มนักเรียนตามแผนกการศึกษา

ฟังก์ชันการทำรายงานภายใน ReportClass มีความยืดหยุ่นสูง ทำให้ผู้ใช้สามารถเลือกการแสดงข้อมูลได้ตามความต้องการ ไม่ว่าจะเป็นการดึงข้อมูลเฉพาะเจาะจง เช่น รายงานนักเรียนที่ลงทะเบียนในปีการศึกษาที่กำหนด หรือการสร้างรายงานแบบครอบคลุมข้อมูลนักเรียนทั้งหมดที่มีในระบบ นอกจากนี้ยังมีฟังก์ชันที่ช่วยในการจัดทำสถิติสำคัญต่าง ๆ เช่น การนับจำนวนนักเรียนในแต่ละปีหรือแผนก การแสดงรายชื่อนักเรียนที่มีครูดูแลเป็นรายบุคคล ซึ่งช่วยให้การจัดการข้อมูลทางการศึกษาเป็นไปอย่างราบรื่นและมีประสิทธิภาพ

นอกจากนี้ ReportClass ยังมีฟังก์ชันที่ออกแบบมาเพื่อบันทึกรายงานในรูปแบบไฟล์ที่สามารถนำไปใช้ในงานเอกสารหรือการนำเสนอข้อมูลได้อย่างสะดวก ผู้ใช้สามารถเลือกได้ว่าจะบันทึก

รายงานในรูปแบบใด เช่น การบันทึกเป็นไฟล์ข้อความ (.txt) หรือไฟล์อื่น ๆ ที่รองรับ ซึ่งทำให้การจัดการข้อมูลที่ต้องการนำไปใช้ภายนอกโปรแกรมเป็นไปอย่างง่ายดาย และสามารถนำข้อมูลเหล่านี้ไปใช้งานในบริบทที่หลากหลาย เช่น การประชุมผู้บริหารหรือการทำรายงานประจำปี

ReportClass ยังมีการจัดการข้อผิดพลาดและการตรวจสอบความสมบูรณ์ของข้อมูลที่สำคัญ เช่น การตรวจสอบว่าข้อมูลในไฟล์ไม่ถูกทำลาย หรือการแจ้งเตือนเมื่อไม่มีข้อมูลเพียงพอสำหรับการสร้างรายงาน ซึ่งช่วยลดความผิดพลาดที่อาจเกิดขึ้นจากการประมวลผลข้อมูลขนาดใหญ่ นอกจากนี้ยังมีการเพิ่มความสามารถในการอัปเดตรายงานได้อย่างต่อเนื่องในกรณีที่มีการเปลี่ยนแปลงข้อมูลนักเรียนหรือครูในระบบ ทำให้ข้อมูลในรายงานเป็นปัจจุบันและถูกต้องเสมอ

ด้วยคุณสมบัติที่หลากหลายและครอบคลุม ReportClass จึงเป็นส่วนสำคัญในการบริหารจัดการข้อมูลการศึกษา ซึ่งช่วยให้ผู้ใช้สามารถสร้างรายงานที่มีประสิทธิภาพและพร้อมใช้งานในสถานการณ์ต่าง ๆ

3.3.1 การกำหนดค่าเริ่มต้น

ในฟังก์ชัน __init__ จะมีการกำหนดตัวแปรภายในคลาสเพื่อเก็บข้อมูลนักเรียนและครู โดย self.students จะถูกใช้เพื่อเก็บลิสต์ของวัตถุที่เป็นนักเรียน และ self.teachers จะเก็บลิสต์ของวัตถุที่เป็นครู ทั้งสองลิสต์นี้จะเริ่มต้นเป็นลิสต์ว่าง ซึ่งจะถูกเติมข้อมูลในฟังก์ชัน load_data

```
class ReportClass:
    def __init__(self):
        self.students = []
        self.teachers = []
```

ภาพที่ 3-14 การกำหนดค่าเริ่มต้น

3.3.2 การโหลดข้อมูล

ฟังก์ชัน load_data จะทำหน้าที่โหลดข้อมูลนักเรียนและครูจากไฟล์ไบนารี student.bin และ teacher.bin โดยใช้การอ่านข้อมูลแบบไบน์ ซึ่งช่วยให้การจัดการข้อมูลที่มีขนาดใหญ่ทำได้อย่างรวดเร็ว ข้อมูลนักเรียนแต่ละคนมีขนาด 92 ไบต์ และข้อมูลครูมีขนาด 63 ไบต์

ในขั้นตอนนี้ จะทำการนำเข้าคลาส StudentClass และ TeacherClass ที่เกี่ยวข้อง ซึ่งช่วยในการสร้างวัตถุสำหรับนักเรียนและครู การอ่านข้อมูลจากไฟล์จะถูกทำในลูป while จนกว่าจะไม่มีข้อมูลเหลืออยู่ โดยเมื่อได้ข้อมูลแล้ว จะมีการเรียกใช้งานฟังก์ชัน unpack ของวัตถุแต่ละประเภทเพื่อทำการแยกข้อมูลออกมาเก็บไว้ในอ็อบเจ็กต์ที่สร้างขึ้น และสุดท้ายจะเพิ่มอ็อบเจ็กต์เหล่านั้นลงในลิสต์ที่กำหนดไว้

```
def load_data(self):
    from student import StudentClass
    from teacher import TeacherClass
    with open('student.bin', 'rb') as file:
        data = file.read()
    while data:
        student = StudentClass(0, '', 0, '', 0)
        student.unpack(data[:92])
        self.students.append(student)
        data = data[92:]
    with open('teacher.bin', 'rb') as file:
        data = file.read()
    while data:
        teacher = TeacherClass(0, '')
        teacher.unpack(data[:63])
        self.teachers.append(teacher)
        data = data[63:]
```

ภาพที่ 3-15 การโหลดข้อมูล

3.3.3 การสร้างรายงานเกี่ยวกับครู

ฟังก์ชัน report_teacher จะสร้างรายงานที่แสดงจำนวนของนักเรียนที่แต่ละครูดูแลอยู่ โดยใช้ลูปในการตรวจสอบทุกครูในลิสต์ self.teachers และสำหรับแต่ละครู จะทำการนับจำนวนของนักเรียนที่มี teacher_id ตรงกัน ซึ่งเป็นข้อมูลที่เก็บไว้ในวัตถุ student ของแต่ละนักเรียน

รายงานนี้มีลักษณะการจัดรูปแบบที่ชัดเจน โดยเริ่มจากชื่อครูและตามด้วยจำนวนของนักเรียนที่อยู่ภายใต้การดูแลของเขา การสร้างรายงานนี้ช่วยให้เห็นภาพรวมของการกระจายของนักเรียนในแต่ละห้องเรียน และทำให้ผู้ดูแลระบบสามารถวิเคราะห์ได้ง่าย

```
def report_teacher(self):
    report_text = ''
    report_text += 'The teacher has students:\n'
    for teacher in self.teachers:
        count = 0
        for student in self.students:
            if student.teacher_id == teacher.teacher_id:
                count += 1
        report_text += f'\t{teacher.name}: {count}\n'
    return report_text
```

ภาพที่ 3-16 การสร้างรายงานเกี่ยวกับครู

3.3.4 การสร้างรายงานตามปีการศึกษา

ฟังก์ชัน report_year มีการนับจำนวนของนักเรียนที่ลงทะเบียนในแต่ละปีการศึกษา โดยใช้ดิกชันนารี years เพื่อเก็บข้อมูลจำนวนของนักเรียนในแต่ละปี นอกจากนั้นยังช่วยให้สามารถเห็นการกระจายของนักเรียนตามปีการศึกษา

การจัดกลุ่มนักเรียนตามปีการศึกษาเป็นวิธีที่ดีในการวิเคราะห์และวางแผนการเรียนการสอน รายงานที่ได้จะทำให้เห็นจำนวนของนักเรียนที่ลงทะเบียนในแต่ละปีการศึกษา ซึ่งสามารถใช้ในการตัดสินใจด้านการจัดการทรัพยากรทางการศึกษา

```
def report_year(self):
    report_text = ''
    years = {}
    for student in self.students:
        if student.year not in years:
            years[student.year] = 1
        else:
            years[student.year] += 1
    report_text += 'Number of students in each year:\n'
    for year in years:
        report_text += f'\tYear {year}: {years[year]}\n'
    return report_text
```

ภาพที่ 3-17 การสร้างรายงานตามปีการศึกษา

3.3.5 การสร้างรายงานตามแผนก

การศึกษา โดยจะมีการใช้ดิกชันนารี departments เพื่อเก็บจำนวนของนักเรียนในแต่ละแผนก การสร้างรายงานในลักษณะนี้จะช่วยให้ผู้บริหารสามารถเข้าใจการกระจายของนักเรียนในแต่ละสาขาวิชาได้อย่างชัดเจน

รายงานนี้ช่วยให้สามารถดูข้อมูลที่เกี่ยวกับการลงทะเบียนของนักเรียนในแต่ละแผนก ซึ่งจะเป็นข้อมูลที่มีประโยชน์ในการจัดการการเรียนการสอนและการพัฒนาหลักสูตร

```
def report_department(self):
    report_text = ''
    departments = {}
    for student in self.students:
        if student.department not in departments:
            departments[student.department] = 1
        else:
            departments[student.department] += 1
    report_text += 'Number of students in each department:\n'
    for department in departments:
        report_text += f'\t{department}: {departments[department]}\n'
    return report_text
```

ภาพที่ 3-18 การสร้างรายงานตามแผนก

3.3.6 การรายงานจำนวนรวมของนักเรียน

ฟังก์ชัน report_total ใช้ในการคืนค่าจำนวนรวมของนักเรียนในระบบ โดยใช้ฟังก์ชัน len() เพื่อให้ได้จำนวนของนักเรียนที่อยู่ในลิสต์ self.students ข้อมูลนี้มีความสำคัญในการวิเคราะห์ขนาดของการศึกษาหรือความต้องการทรัพยากรในโรงเรียน

```
def report_total(self):
    return f'Total number of students:\t{len(self.students)}'
```

ภาพที่ 3-19 การรายงานจำนวนรวมของนักเรียน

3.3.7 การบันทึกรายงาน

ฟังก์ชัน save_report มีหน้าที่ในการรวบรวมข้อมูลทั้งหมดที่ได้จากการสร้างรายงาน และบันทึกลงในไฟล์ report.txt โดยจะเรียกใช้งานฟังก์ชัน load_data เพื่อให้แน่ใจว่าข้อมูลที่ถูกโหลดล่าสุดจะถูกนำไปใช้ในรายงาน

การบันทึกข้อมูลในไฟล์จะทำให้ง่ายต่อการเก็บรักษาและสามารถใช้ในการวิเคราะห์ในอนาคต ผลลัพธ์จะถูกจัดรูปแบบให้ง่ายต่อการอ่าน โดยเริ่มจากหัวข้อรายงาน ตามด้วยข้อมูลที่จัดเก็บในรายงานที่เกี่ยวข้องกับครู ปีการศึกษา แผนก และจำนวนรวมของนักเรียน

```
def save_report(self):
    with open('report.txt', 'w') as file:
        self.load_data()
        file.write("Student Survey Report\n")
        file.write(self.report_teacher())
        file.write(self.report_year())
        file.write(self.report_department())
        file.write(self.report_total())
        print("Report saved to report.txt")
```

ภาพที่ 3-20 การบันทึกรายงาน

คลาส ReportClass ในไฟล์ report.py ถูกออกแบบมาเพื่อให้สามารถจัดการและสร้างรายงานที่สำคัญเกี่ยวกับนักเรียนและครูในระบบการศึกษาได้อย่างมีประสิทธิภาพ การโหลดข้อมูล การสร้างรายงาน และการบันทึกข้อมูลทั้งหมดนี้เป็นเครื่องมือที่ช่วยในการวิเคราะห์และการตัดสินใจในด้านการศึกษาที่มีความซับซ้อน ฟังก์ชันที่มีอยู่ในคลาสนี้ทำให้สามารถเข้าถึงข้อมูลที่สำคัญได้อย่างรวดเร็วและมีประสิทธิภาพ โดยเฉพาะในการบริหารจัดการการศึกษาและการวางแผนการพัฒนาหลักสูตรที่ตอบสนองความต้องการของนักเรียนและครู

3.4 ไฟล์ student.py

ไฟล์ student.py นี้ออกแบบคลาส StudentClass เพื่อให้รองรับการจัดการข้อมูลนักเรียนในระบบการศึกษาได้อย่างมีประสิทธิภาพ โดยมีการสร้างฟังก์ชันหลากหลายเพื่อรองรับการจัดการ

ข้อมูลในหลายมิติ เช่น การเพิ่มข้อมูล การแก้ไขข้อมูล การลบข้อมูล รวมถึงการดึงข้อมูลนักเรียนในรูปแบบต่าง ๆ ตามเงื่อนไขที่กำหนด เช่น ค้นหานักเรียนตามหมายเลขประจำตัว ค้นหานักเรียนตามชื่อ ค้นหานักเรียนตามปีการศึกษา แผนกการศึกษา และครูที่รับผิดชอบ ซึ่งทั้งหมดนี้ถูกออกแบบเพื่อให้การจัดการข้อมูลนักเรียนเป็นไปอย่างมีระบบและง่ายต่อการใช้งาน

ฟังก์ชันการทำงานที่ครอบคลุม

คลาส StudentClass มีฟังก์ชันที่ครอบคลุมความต้องการในการจัดการข้อมูลนักเรียน เช่น ฟังก์ชัน add_student ทำหน้าที่เพิ่มข้อมูลนักเรียนใหม่ลงในระบบ โดยฟังก์ชันนี้จะรับข้อมูลที่จำเป็นจากผู้ใช้ ตรวจสอบความถูกต้องของข้อมูลผ่านฟังก์ชัน validation ซึ่งจะตรวจสอบความถูกต้องของข้อมูลที่ส่งเข้ามา เพื่อให้มั่นใจว่าข้อมูลที่เพิ่มลงในระบบมีความสมบูรณ์และถูกต้อง จากนั้นจะทำการบันทึกข้อมูลนักเรียนลงในไฟล์ในรูปแบบไบนารี เพื่อให้ประหยัดเนื้อที่และจัดการข้อมูลได้ง่าย

การลบข้อมูลนักเรียนสามารถทำได้ผ่านฟังก์ชัน remove_student ซึ่งจะอ่านข้อมูลนักเรียนทั้งหมดจากไฟล์ student.bin จากนั้นจะตรวจสอบหมายเลขประจำตัวนักเรียนที่ต้องการลบ และลบข้อมูลนั้นออกจากไฟล์โดยไม่เขียนกลับไปในไฟล์อีก การอัปเดตข้อมูลนักเรียนก็ทำงานในลักษณะคล้ายกัน โดยฟังก์ชัน update_student จะตรวจสอบความถูกต้องของข้อมูลก่อน และหากข้อมูลถูกต้องก็จะทำการอัปเดตข้อมูลใหม่ลงในไฟล์ทดแทนข้อมูลเก่า

การดึงข้อมูลแบบหลากหลายมิติ

คลาสนี้ยังมีฟังก์ชันที่ออกแบบมาเพื่อรองรับการดึงข้อมูลนักเรียนตามเงื่อนไขที่หลากหลาย เช่น ฟังก์ชัน get_student_by_id ใช้เพื่อค้นหานักเรียนโดยใช้หมายเลขประจำตัวนักเรียน ซึ่งเป็นการดึงข้อมูลที่เป็นแบบเจาะจงและเฉพาะเจาะจงต่อบุคคล นอกจากนี้ยังมีฟังก์ชันอย่าง get_student_by_name ซึ่งใช้ในการค้นหานักเรียนโดยใช้ชื่อ หรือฟังก์ชัน get_student_by_year ที่ค้นหานักเรียนตามปีการศึกษา และฟังก์ชัน get_student_by_department ที่ใช้ดึงข้อมูลนักเรียนตามแผนกการศึกษาที่ลงทะเบียน ทั้งหมดนี้ทำให้สามารถค้นหานักเรียนได้ตามความต้องการที่หลากหลาย และสะดวกต่อการนำข้อมูลไปใช้งานในบริบทที่ต่างกัน

การจัดเก็บข้อมูลในรูปแบบไบนารี

ข้อมูลของนักเรียนทั้งหมดถูกจัดเก็บในรูปแบบไบนารี โดยใช้โมดูล struct ซึ่งช่วยให้สามารถบันทึกและดึงข้อมูลกลับมาได้อย่างมีประสิทธิภาพ รูปแบบไบนารีทำให้ไฟล์มีขนาดเล็กและประหยัด

เนื้อที่ในการจัดเก็บ นอกจากนี้ยังสามารถเรียกคืนข้อมูลได้อย่างรวดเร็ว ฟังก์ชัน pack และ unpack ทำหน้าที่สำคัญในการแปลงข้อมูลนักเรียนจากรูปแบบที่มนุษย์อ่านเข้าใจได้ให้กลายเป็นไบนารี และแปลงกลับมาเป็นรูปแบบที่สามารถอ่านได้อีกครั้งเมื่อมีการเรียกดูข้อมูล

การตรวจสอบความถูกต้องของข้อมูล

หนึ่งในองค์ประกอบสำคัญของการออกแบบคลาสนี้คือการตรวจสอบความถูกต้องของข้อมูลก่อนที่จะทำการบันทึกหรืออัปเดตลงในไฟล์ ฟังก์ชัน validation จะตรวจสอบข้อมูลนักเรียน เช่น ตรวจสอบว่าชื่อไม่ยาวเกิน 50 ตัวอักษร ตรวจสอบปีการศึกษาและแผนกการศึกษาว่ามีอยู่ในระบบหรือไม่ นอกจากนี้ยังตรวจสอบว่าหมายเลขประจำตัวของนักเรียนและครูถูกต้องหรือไม่ หากมีความผิดพลาดในข้อมูลจะมีการแสดงข้อความแจ้งเตือน เพื่อให้ผู้ใช้สามารถแก้ไขข้อมูลได้อย่างถูกต้องก่อนทำการบันทึกลงในระบบ

การออกแบบเพื่อรองรับการขยายระบบในอนาคต

คลาส StudentClass ถูกออกแบบมาให้สามารถขยายการทำงานในอนาคตได้ง่าย เนื่องจากมีการจัดการข้อมูลผ่านฟังก์ชันที่เป็นอิสระและมีโครงสร้างที่ชัดเจน ทำให้สามารถเพิ่มเติมฟังก์ชันใหม่ ๆ ได้ เช่น การจัดการข้อมูลนักเรียนที่มีหลายคลาสเรียนหรือการเพิ่มข้อมูลเกี่ยวกับผลการเรียนเข้าไปในระบบ โดยที่ไม่จำเป็นต้องแก้ไขโครงสร้างหลักของคลาส

student.py เป็นระบบที่ออกแบบมาอย่างมีประสิทธิภาพเพื่อให้รองรับการจัดการข้อมูลนักเรียนในหลากหลายรูปแบบ มีการตรวจสอบความถูกต้องของข้อมูลก่อนที่จะทำการบันทึกหรืออัปเดต มีฟังก์ชันที่หลากหลายในการดึงข้อมูลตามเงื่อนไขต่าง ๆ ทำให้ระบบมีความยืดหยุ่นและพร้อมสำหรับการขยายในอนาคต

3.4.1 การนำเข้าโมดูล

ในส่วนนี้ มีการนำเข้าโมดูลที่จำเป็นสำหรับการทำงานของคลาส StudentClass โดย struct จะใช้สำหรับการแปลงข้อมูลระหว่างรูปแบบไบนารีและรูปแบบที่เข้าใจได้ง่าย ในขณะที่ os จะใช้สำหรับการตรวจสอบการมีอยู่ของไฟล์ นอกจากนี้ยังนำเข้า YEARS และ DEPARTMENTS จากไฟล์ enum_1.py ซึ่งมีค่าที่กำหนดไว้ล่วงหน้า เช่น ปีการศึกษาและแผนกการศึกษา

```
import struct
import os
from enum_1 import YEARS, DEPARTMENTS
```

ภาพที่ 3-21 การนำเข้าโมดูล

3.4.2 **คลาส** `StudentClass`

คลาส `StudentClass` นี้เป็นตัวแทนของนักเรียน ซึ่งมีฟังก์ชันและตัวแปรที่เกี่ยวข้องกับข้อมูลนักเรียน

```
class StudentClass:
```

ภาพที่ 3-22 คลาส StudentClass

3.4.3 ฟังก์ชัน __init__

ฟังก์ชัน __init__ ทำหน้าที่เป็น constructor ของคลาส StudentClass โดยจะถูกเรียกใช้เมื่อมีการสร้างอ็อบเจ็กต์ใหม่ของนักเรียน ฟังก์ชันนี้จะกำหนดค่าต่าง ๆ ให้กับอ็อบเจ็กต์ที่สร้างขึ้น เช่น หมายเลขประจำตัวนักเรียน (student_id), ชื่อ (name), ปีการศึกษา (year), แผนกการศึกษา (department), และหมายเลขประจำตัวของครูที่ดูแลนักเรียน (teacher_id). ข้อมูลเหล่านี้จะถูกเก็บไว้ในตัวแปรภายในคลาส เพื่อให้สามารถใช้งานได้ในฟังก์ชันอื่น ๆ ที่เกี่ยวข้องกับข้อมูลนักเรียน

```
def __init__(self, student_id, name, year, department, teacher_id):
```

ภาพที่ 3-23 ฟังก์ชัน __init__

3.4.4 ฟังก์ชัน pack

ฟังก์ชัน pack ทำหน้าที่แปลงข้อมูลของนักเรียนในอ็อบเจ็กต์ให้เป็นรูปแบบไบนารีที่สามารถบันทึกลงในไฟล์ได้ โดยใช้ struct.pack เพื่อจัดระเบียบข้อมูลในรูปแบบที่กำหนด เช่น หมายเลขประจำตัวนักเรียนจะถูกจัดเก็บเป็นจำนวนเต็ม และชื่อจะถูกแปลงเป็นไบต์ การแปลงนี้ช่วยให้การจัดเก็บข้อมูลมีขนาดที่เล็กลงและเป็นไปตามโครงสร้างที่สามารถอ่านได้ง่ายในรูปแบบไฟล์ไบนารี

```
def pack(self):
```

ภาพที่ 3-24 ฟังก์ชัน pack

3.4.5 ฟังก์ชัน unpack

ฟังก์ชัน unpack ใช้สำหรับการแปลงข้อมูลที่ถูกอ่านจากไฟล์ในรูปแบบไบนารีกลับมาเป็นข้อมูลที่สามารถเข้าใจได้ โดยใช้ struct.unpack เพื่อแยกข้อมูลในรูปแบบไบนารีออกเป็นค่าต่าง ๆ ซึ่งจะถูกจัดเก็บในตัวแปรภายในของคลาส ฟังก์ชันนี้ช่วยให้สามารถดึงข้อมูลของนักเรียนออกมาได้จากไฟล์และนำไปใช้งานในโปรแกรมต่อไป

```
def unpack(self, data):
```

ภาพที่ 3-25 ฟังก์ชัน unpack

3.4.6 ฟังก์ชัน validation

ฟังก์ชัน validation ทำหน้าที่ตรวจสอบความถูกต้องของข้อมูลนักเรียนก่อนที่จะบันทึกหรืออัปเดตข้อมูลลงในไฟล์ ฟังก์ชันนี้จะตรวจสอบว่าข้อมูลต่าง ๆ เช่น ชื่อปีการศึกษาและแผนกการศึกษาอยู่ในรูปแบบที่ถูกต้องหรือไม่ โดยจะมีการตรวจสอบว่าชื่อไม่ยาวเกินกว่าที่กำหนด ID ของนักเรียนและครูต้องมากกว่า 0 และต้องตรวจสอบว่าปีการศึกษาตรงตามที่กำหนดใน YEARS และแผนกการศึกษาต้องอยู่ใน DEPARTMENTS หากข้อมูลไม่ถูกต้อง จะมีการพิมพ์ข้อความแจ้งเตือน

```
def validation(self, is_update=False):
```

ภาพที่ 3-26 ฟังก์ชัน validation

3.4.7 ฟังก์ชัน add_student

ฟังก์ชัน add_student ทำหน้าที่เพิ่มข้อมูลนักเรียนใหม่ลงในไฟล์ student.bin โดยจะเรียกใช้ฟังก์ชัน validation เพื่อตรวจสอบความถูกต้องของข้อมูลก่อน หากข้อมูลถูกต้อง โปรแกรมจะเปิดไฟล์ในโหมด append binary (ab) และเขียนข้อมูลนักเรียนใหม่ลงไป โดยข้อมูลจะถูกแปลงเป็น

รูปแบบไบนารีผ่านฟังก์ชัน pack. หากไฟล์ไม่อยู่ จะทำการสร้างไฟล์ใหม่และเขียนข้อมูลลงไป การเพิ่มนักเรียนจะเสร็จสมบูรณ์ด้วยการพิมพ์ข้อความยืนยัน

```
def add_student(self):
```

ภาพที่ 3-27 ฟังก์ชัน add_student

3.4.8 ฟังก์ชัน remove_student

ฟังก์ชัน remove_student มีหน้าที่ลบข้อมูลนักเรียนจากไฟล์ student.bin โดยการอ่านข้อมูลทั้งหมดจากไฟล์และตรวจสอบหมายเลขประจำตัวนักเรียนที่ต้องการลบ ถ้าพบว่านักเรียนที่มีหมายเลขประจำตัวตรงกัน จะไม่เขียนข้อมูลนั้นกลับไปยังไฟล์ ซึ่งจะทำให้ข้อมูลของนักเรียนถูกลบออกไปจากไฟล์อย่างถาวร

```
def remove_student(self, student_id):
```

ภาพที่ 3-28 ฟังก์ชัน remove_student

3.4.9 ฟังก์ชัน update_student

ฟังก์ชัน update_student ทำหน้าที่อัปเดตข้อมูลของนักเรียนในไฟล์ โดยจะเริ่มต้นด้วยการตรวจสอบความถูกต้องของข้อมูลด้วยฟังก์ชัน validation หากข้อมูลถูกต้อง โปรแกรมจะอ่านข้อมูลทั้งหมดจากไฟล์และเปรียบเทียบหมายเลขประจำตัวนักเรียน ถ้าพบหมายเลขที่ตรงกัน จะทำการเขียนข้อมูลใหม่ลงไปในไฟล์แทนที่ข้อมูลเก่า ซึ่งช่วยให้ข้อมูลนักเรียนเป็นปัจจุบันและถูกต้อง

```
def update_student(self):
```

ภาพที่ 3-29 ฟังก์ชัน update_student

3.4.10 ฟังก์ชัน get_student_by_id

ฟังก์ชัน get_student_by_id ใช้ในการค้นหานักเรียนโดยใช้หมายเลขประจำตัวนักเรียน โดยการอ่านข้อมูลจากไฟล์และเปรียบเทียบหมายเลขประจำตัวนักเรียนในแต่ละรายการ ถ้าพบจะส่งคืนอ็อบเจ็กต์ของนักเรียนที่ตรงกัน ซึ่งช่วยให้สามารถดึงข้อมูลนักเรียนได้อย่างรวดเร็วตาม ID

```
def get_student_by_id(self, student_id):
```

ภาพที่ 3-30 ฟังก์ชัน get_student_by_id

3.4.11 ฟังก์ชัน get_student_by_name

ฟังก์ชัน get_student_by_name ทำหน้าที่ค้นหานักเรียนตามชื่อ โดยอ่านข้อมูลจากไฟล์และเปรียบเทียบชื่อของนักเรียนในแต่ละรายการ ถ้าชื่อตรงกันจะส่งคืนอ็อบเจ็กต์ของนักเรียนที่พบ ซึ่งช่วยให้การค้นหานักเรียนตามชื่อมีความสะดวกและรวดเร็ว

```
def get_student_by_name(self, name):
```

ภาพที่ 3-31 ฟังก์ชัน get_student_by_name

3.4.12 ฟังก์ชัน get_student_by_year

ฟังก์ชัน get_student_by_year ใช้ในการค้นหานักเรียนตามปีการศึกษา โดยจะสร้างลิสต์เพื่อเก็บนักเรียนที่อยู่ในปีการศึกษานั้น ๆ โดยอ่านข้อมูลจากไฟล์และตรวจสอบปีการศึกษา หากปีการศึกษาตรงกันจะเพิ่มนักเรียนลงในลิสต์ที่สร้างขึ้น ซึ่งจะช่วยให้สามารถดึงข้อมูลนักเรียนตามปีการศึกษาได้อย่างมีประสิทธิภาพ

```
def get_student_by_year(self, year):
```

ภาพที่ 3-32 ฟังก์ชัน get_student_by_year

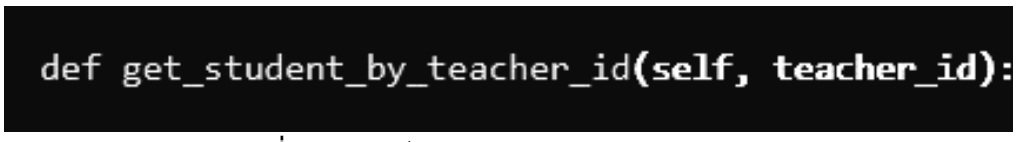
3.4.13 ฟังก์ชัน get_student_by_department

ฟังก์ชัน get_student_by_department ทำหน้าที่ค้นหานักเรียนตามแผนกการศึกษา โดยจะทำการอ่านข้อมูลจากไฟล์และตรวจสอบแผนกการศึกษาในแต่ละรายการ หากตรงกันจะเพิ่มนักเรียนลงในลิสต์ที่สร้างขึ้น ซึ่งทำให้การดึงข้อมูลนักเรียนตามแผนกการศึกษาเป็นเรื่องง่ายและรวดเร็ว

ภาพที่ 3-33 ฟังก์ชัน get_student_by_department

3.4.14 ฟังก์ชัน get_student_by_teacher_id

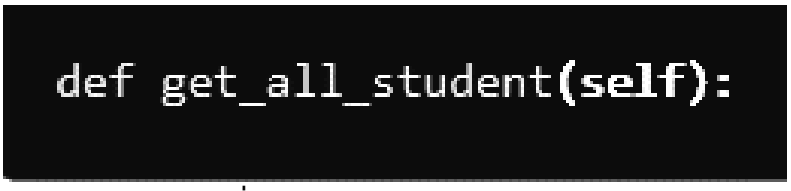
ฟังก์ชัน get_student_by_teacher_id ใช้สำหรับค้นหานักเรียนตามหมายเลขประจำตัวของครู โดยจะอ่านข้อมูลจากไฟล์และตรวจสอบว่ามีนักเรียนที่มี teacher_id ตรงกัน ถ้าพบจะเพิ่มนักเรียนลงในลิสต์ที่สร้างขึ้น ซึ่งช่วยให้การดึงข้อมูลนักเรียนที่ถูกดูแลโดยครูคนเดียวกันเป็นไปได้ง่าย



ภาพที่ 3-34 ฟังก์ชัน get_student_by_teacher_id

3.4.15 ฟังก์ชัน get_all_student

ฟังก์ชัน get_all_student ใช้เพื่อดึงข้อมูลนักเรียนทั้งหมดจากไฟล์ student.bin โดยจะอ่านข้อมูลทั้งหมดและสร้างลิสต์ของนักเรียนทั้งหมดในไฟล์ ซึ่งทำให้สามารถดึงข้อมูลนักเรียนทั้งหมดออกมาได้ในครั้งเดียว ซึ่งมีความสะดวกและมีประสิทธิภาพในกรณีที่ต้องการดูข้อมูลนักเรียนทั้งหมด



ภาพที่ 3-35 ฟังก์ชัน get_all_student

โดยรวมแล้ว StudentClass เป็นคลาสที่ออกแบบมาเพื่อติดต่อกับข้อมูลนักเรียนในระบบการศึกษา โดยมีฟังก์ชันที่รองรับการเพิ่ม ลบ อัปเดต ค้นหา และดึงข้อมูลนักเรียนจากไฟล์ในรูปแบบไบนารี การใช้โมดูล struct ช่วยให้สามารถจัดการข้อมูลที่มีโครงสร้างแน่นอนได้อย่างมีประสิทธิภาพ

3.5 ไฟล์ teacher.py

ไฟล์ teacher.py ถูกออกแบบมาเพื่อจัดการข้อมูลครูในระบบการศึกษา โดยมีการใช้งานคลาส TeacherClass ซึ่งมีหน้าที่หลักในการบันทึก อัปเดต ลบ และดึงข้อมูลครูที่เก็บไว้ในไฟล์แบบไบต์ (teacher.bin) การจัดเก็บข้อมูลในรูปแบบไบต์นั้นเหมาะสมอย่างยิ่งสำหรับการประมวลผลข้อมูลที่มีขนาดใหญ่ และช่วยให้การเข้าถึงข้อมูลทำได้อย่างรวดเร็วและมีประสิทธิภาพ

การจัดการข้อมูลครู

ภายในคลาส TeacherClass มีการนิยามฟังก์ชันต่าง ๆ ที่ใช้ในการจัดการข้อมูลครู ซึ่งรวมถึงฟังก์ชันสำหรับเพิ่มข้อมูลใหม่ (add_teacher), อัปเดตข้อมูลที่มีอยู่ (update_teacher), ลบข้อมูลที่ไม่ต้องการ (remove_teacher), และดึงข้อมูลทั้งหมดหรือเฉพาะเจาะจง (get_all_teachers, get_teacher_by_id, get_teacher_by_name) การออกแบบฟังก์ชันเหล่านี้ช่วยให้ผู้ใช้สามารถดำเนินการจัดการข้อมูลครูได้อย่างคล่องตัวและมีประสิทธิภาพ

การจัดเก็บข้อมูลแบบไบต์

การใช้ไฟล์แบบไบต์ (teacher.bin) ในการจัดเก็บข้อมูลครูทำให้สามารถประมวลผลข้อมูลได้เร็วขึ้นเมื่อเปรียบเทียบกับการจัดเก็บในรูปแบบข้อความหรือ CSV ซึ่งข้อมูลจะถูกจัดเก็บในรูปแบบที่สามารถจัดการได้อย่างมีประสิทธิภาพผ่านการใช้โมดูล struct การใช้โมดูลนี้ช่วยให้สามารถแปลงข้อมูลให้เป็นไบต์และเรียกคืนข้อมูลจากไบต์ให้อยู่ในรูปแบบที่สามารถเข้าใจได้ เช่น การจัดเก็บข้อมูลในรูปแบบของหมายเลขประจำตัวครู ชื่อ นามสกุล และข้อมูลอื่น ๆ ที่เกี่ยวข้อง

การเพิ่มข้อมูลครู

ฟังก์ชัน add_teacher ถูกออกแบบมาเพื่อเพิ่มข้อมูลครูใหม่ลงในไฟล์ teacher.bin โดยก่อนที่จะทำการบันทึกข้อมูล ระบบจะมีการตรวจสอบความถูกต้องของข้อมูลที่ป้อนเข้ามา เช่น

ตรวจสอบว่าหมายเลขประจำตัวครูมีความซ้ำซ้อนหรือไม่ เพื่อป้องกันการบันทึกข้อมูลที่ไม่ถูกต้องหรือซ้ำซ้อน

การอัปเดตและลบข้อมูลครู

ในกรณีที่ต้องการอัปเดตข้อมูลของครูที่มีอยู่แล้ว ฟังก์ชัน update_teacher จะทำการค้นหาข้อมูลครูตามหมายเลขประจำตัว และอนุญาตให้ผู้ใช้สามารถแก้ไขข้อมูลที่ต้องการได้ หลังจากที่ทำการแก้ไขแล้ว ข้อมูลที่อัปเดตจะถูกเขียนทับลงในไฟล์ teacher.bin เพื่อให้แน่ใจว่าข้อมูลล่าสุดจะถูกเก็บรักษาไว้ นอกจากนี้ ฟังก์ชัน remove_teacher ยังช่วยให้ผู้ใช้สามารถลบข้อมูลครูที่ไม่ต้องการออกจากระบบได้อย่างง่ายดาย โดยทำการค้นหาและลบข้อมูลที่ตรงตามหมายเลขประจำตัวที่กำหนด

การดึงข้อมูลครู

การดึงข้อมูลครูที่เก็บไว้ในระบบสามารถทำได้อย่างสะดวกผ่านฟังก์ชัน get_all_teachers ซึ่งจะส่งคืนข้อมูลของครูทั้งหมดในรูปแบบที่สามารถอ่านได้ นอกจากนี้ยังมีฟังก์ชัน get_teacher_by_id และ get_teacher_by_name ที่ช่วยให้ผู้ใช้สามารถค้นหาข้อมูลครูตามหมายเลขประจำตัวหรือชื่อตามที่ต้องการ ซึ่งเป็นการเพิ่มความสะดวกในการค้นหาข้อมูลในระบบ

โดยรวมแล้ว ไฟล์ teacher.py เป็นส่วนสำคัญในการจัดการข้อมูลครูในระบบการศึกษา มีการออกแบบที่มุ่งเน้นประสิทธิภาพในการจัดเก็บและจัดการข้อมูลผ่านการใช้เทคนิคการประมวลผลแบบไบต์และโมดูล struct ซึ่งทำให้สามารถจัดการข้อมูลได้อย่างรวดเร็วและมีประสิทธิภาพ ฟังก์ชันต่าง ๆ ที่ถูกออกแบบในคลาส TeacherClass ทำให้ผู้ใช้สามารถดำเนินการจัดการข้อมูลได้อย่างง่ายดาย ไม่ว่าจะเป็นการเพิ่ม อัปเดต ลบ หรือดึงข้อมูลครูตามที่ต้องการ

3.5.1 การนำเข้าโมดูล

ส่วนนี้ทำการนำเข้าโมดูล struct และ os เพื่อช่วยในการจัดการข้อมูลและระบบไฟล์ struct ใช้ในการบีบอัดและถอดรหัสข้อมูลในรูปแบบไบต์ ซึ่งเหมาะสำหรับการเก็บข้อมูลในไฟล์ไบนารี (.bin) os ใช้ตรวจสอบว่ามีไฟล์อยู่หรือไม่ และช่วยในการจัดการไฟล์ในระบบ

```
import struct
import os
```

ภาพที่ 3-36 การนำเข้าโมดูล

3.5.2 คลาส TeacherClass

คลาส TeacherClass ถูกสร้างขึ้นเพื่อเก็บข้อมูลเกี่ยวกับครู โดยมีสองฟิลด์หลัก teacher_id: รหัสครู (เป็นค่าจำนวนเต็ม) name: ชื่อของครู (เป็นค่าข้อความ)

```
class TeacherClass:
    def __init__(self, teacher_id, name):
        self.teacher_id = teacher_id
        self.name = name
```

ภาพที่ 3-37 คลาส TeacherClass

3.5.3 การทำงานของ pack และ unpack

การถอดรหัสข้อมูลกลับมาในรูปแบบที่อ่านได้ ฟังก์ชัน pack ทำหน้าที่แปลงรหัสครูและชื่อครูให้อยู่ในรูปแบบไบต์โดยใช้โครงสร้างที่กำหนด เช่น 13s50s หมายถึง รหัสครูใช้ 13 ไบต์ และชื่อครูใช้ 50 ไบต์ การบีบอัดข้อมูลนี้จะทำให้การจัดเก็บในไฟล์ใช้พื้นที่น้อยลง ฟังก์ชัน unpack จะทำการถอดรหัสไบต์ที่เก็บในไฟล์กลับมาเป็นข้อมูลครู โดยการล้างตัวอักษรพิเศษที่ไม่จำเป็นเช่น \x00 ซึ่งเป็นค่าที่ถูกเติมในระหว่างการบีบอัด

```
def pack(self):
    return struct.pack('13s50s', str(self.teacher_id).encode(), self.name.encode())

def unpack(self, data):
    unpacked = struct.unpack('13s50s', data)
    self.teacher_id = int(unpacked[0].decode().strip('\x00'))
    self.name = unpacked[1].decode().strip('\x00')
```

ภาพที่ 3-38 การทำงานของ pack และ unpack

3.5.4 ฟังก์ชัน validation

ฟังก์ชันนี้ถูกใช้เพื่อทำการตรวจสอบความถูกต้องของข้อมูลครูก่อนที่จะบันทึกหรืออัปเดตข้อมูลลงในไฟล์ ฟังก์ชันจะตรวจสอบความยาวของชื่อครูว่าไม่เกิน 50 ตัวอักษรและรหัสครูต้องเป็นค่าบวก นอกจากนี้หากเป็นการเพิ่มข้อมูลใหม่ จะมีการตรวจสอบว่ามีครูที่ใช้รหัสหรือตรงกับชื่อนั้นอยู่แล้วหรือไม่ โดยการอ่านข้อมูลจากไฟล์ teacher.bin ทั้งหมด หากมีข้อมูลซ้ำจะไม่อนุญาตให้บันทึกและจะแสดงข้อความข้อผิดพลาดออกมา

```
def validation(self, is_update=False):
```

ภาพที่ 3-39 ฟังก์ชัน validation

3.5.5 การเพิ่มข้อมูล

ฟังก์ชัน add_teacher ทำหน้าที่เพิ่มข้อมูลครูใหม่ลงในไฟล์ teacher.bin หลังจากผ่านการตรวจสอบจากฟังก์ชัน validation หากข้อมูลผ่านการตรวจสอบแล้ว จะทำการบีบอัดข้อมูลครูให้เป็นไบต์ด้วยฟังก์ชัน pack จากนั้นเขียนข้อมูลลงในไฟล์ teacher.bin โดยจะทำการสร้างไฟล์ใหม่หากยังไม่มีไฟล์นี้อยู่ในระบบ

```
def add_teacher(self):
```

ภาพที่ 3-40 ฟังก์ชัน add_teacher

3.5.6 การลบข้อมูล

ฟังก์ชัน remove_teacher ทำหน้าที่ลบข้อมูลครูออกจากระบบโดยจะค้นหารหัสครูที่ตรงกับข้อมูลที่ต้องการลบ จากนั้นจะเขียนข้อมูลครูที่เหลือกลับไปยังไฟล์ teacher.bin ใหม่ ฟังก์ชันนี้ยังทำหน้าที่ลบข้อมูลนักเรียนที่มีครูคนนั้นเป็นผู้ดูแลออกจากไฟล์ student.bin ด้วย โดยการตรวจสอบข้อมูลนักเรียนทุกคนและลบข้อมูลนักเรียนที่มีการเชื่อมโยงกับครูที่ถูกลบ

```
def remove_teacher(self, teacher_id):
```

ภาพที่ 3-41 ฟังก์ชัน remove_teacher

3.5.7 การอัปเดตข้อมูล

ฟังก์ชัน update_teacher จะทำหน้าที่อัปเดตข้อมูลครูที่มีอยู่ในไฟล์ teacher.bin โดยจะทำการตรวจสอบข้อมูลใหม่ผ่านฟังก์ชัน validation หากข้อมูลใหม่ผ่านการตรวจสอบ จะทำการค้นหาครูที่มีรหัสตรงกันในไฟล์และทำการเขียนข้อมูลใหม่ทับข้อมูลเดิมลงในไฟล์

```
def update_teacher(self):
```

ภาพที่ 3-42 ฟังก์ชัน update_teacher

3.5.8 ฟังก์ชันการค้นหาครู

ฟังก์ชัน get_teacher_by_id และ get_teacher_by_name ใช้สำหรับค้นหาข้อมูลครูตามรหัสครูหรือชื่อครูตามลำดับ โดยจะอ่านข้อมูลจากไฟล์ teacher.bin และค้นหาครูที่มีรหัสหรือชื่อที่ตรงกับเงื่อนไข หากพบข้อมูลตรงกันจะส่งคืนข้อมูลครูนั้นให้กับผู้ใช้

```
@staticmethod
def get_teacher_by_id(teacher_id):
@staticmethod
def get_teacher_by_name(name):
```

ภาพที่ 3-43 ฟังก์ชัน get_teacher_by_id และ get_teacher_by_name

3.5.9 การดึงข้อมูลครูทั้งหมด

ฟังก์ชัน get_all_teacher ทำหน้าที่ดึงข้อมูลครูทั้งหมดจากไฟล์ teacher.bin โดยจะทำการอ่านข้อมูลครูแต่ละบล็อกและถอดรหัสข้อมูลกลับมาในรูปแบบที่อ่านได้ จากนั้นจะเก็บข้อมูลครูทั้งหมดในลิสต์และส่งคืนลิสต์นั้นให้กับผู้ใช้

```
@staticmethod
def get_all_teacher():
```

ภาพที่ 3-44 ฟังก์ชัน get_all_teacher

3.6 ไฟล์ student.bin

ไฟล์ student.bin เป็นส่วนสำคัญของระบบการจัดการข้อมูลนักเรียนในโปรแกรมนี้ โดยมีการจัดเก็บข้อมูลที่เกี่ยวข้องกับนักเรียนในรูปแบบไบต์เพื่อให้สามารถจัดการได้อย่างมีประสิทธิภาพ ข้อมูลที่ถูกเก็บในไฟล์นี้ประกอบด้วยฟิลด์ต่าง ๆ เช่น รหัสนักเรียน ชื่อ นามสกุล ปีการศึกษา แผนกการศึกษา และรหัสครูที่ดูแลนักเรียน ซึ่งมีการออกแบบให้ตรงกับโครงสร้างที่กำหนดในคลาส StudentClass

การจัดเก็บข้อมูลในรูปแบบไบต์นั้นมีข้อดีหลายประการ เช่น

ประสิทธิภาพในการจัดเก็บ ข้อมูลที่ถูกบีบอัดในรูปแบบไบต์ช่วยลดพื้นที่ที่ใช้ในการจัดเก็บข้อมูลเมื่อเทียบกับการจัดเก็บในรูปแบบที่มนุษย์อ่านได้ เช่น ไฟล์ข้อความหรือ CSV

ความเร็วในการเข้าถึงข้อมูล โปรแกรมสามารถอ่านและเขียนข้อมูลได้รวดเร็ว เนื่องจากข้อมูลที่ถูกบีบอัดในรูปแบบไบต์จะมีโครงสร้างที่แน่นอนและใช้พื้นที่น้อยลง ทำให้สามารถจัดการกับข้อมูลขนาดใหญ่ได้อย่างมีประสิทธิภาพ

การตรวจสอบความถูกต้อง การใช้ไฟล์แบบไบต์ทำให้สามารถตรวจสอบความถูกต้องของข้อมูลได้อย่างรวดเร็ว โดยการอ่านข้อมูลในแต่ละบล็อกและทำการเปรียบเทียบข้อมูลกับข้อมูลที่ต้องการ

ในส่วนของการจัดการข้อมูล นักเรียนสามารถถูกเพิ่มลงในไฟล์นี้ได้ผ่านฟังก์ชัน add_student ซึ่งจะทำการตรวจสอบข้อมูลก่อนที่จะทำการเขียนข้อมูลลงไฟล์ หากนักเรียนถูกลบออกจากระบบ ฟังก์ชัน remove_student จะทำการค้นหารหัสนักเรียนที่ตรงกันในไฟล์และเขียนข้อมูลที่เหลือกลับไปยังไฟล์ใหม่

3.7 ไฟล์ teacher.bin

ไฟล์ teacher.bin ทำหน้าที่จัดเก็บข้อมูลของครูในระบบการศึกษา ซึ่งมีความสำคัญในการเชื่อมโยงข้อมูลระหว่างครูและนักเรียน ข้อมูลในไฟล์นี้ประกอบด้วยรหัสครู ชื่อ และอาจมีข้อมูลเพิ่มเติมอื่น ๆ ตามที่กำหนดในคลาส TeacherClass การจัดเก็บข้อมูลในรูปแบบไบต์ช่วยให้การประมวลผลและการเข้าถึงข้อมูลมีประสิทธิภาพมากยิ่งขึ้น โดยทำให้การเข้าถึงข้อมูลสามารถทำได้อย่างรวดเร็วและประหยัดพื้นที่ในการจัดเก็บ เมื่อข้อมูลถูกบันทึกในไฟล์นี้แล้ว ระบบจะสามารถทำการค้นหา ดึงข้อมูล เพิ่ม และลบข้อมูลของครูได้อย่างสะดวก ซึ่งเป็นการสร้างความเชื่อมโยงระหว่างข้อมูลครูกับนักเรียน ทำให้การจัดการข้อมูลในระบบการศึกษาเป็นไปอย่างมีประสิทธิภาพและมีความถูกต้อง

การจัดเก็บข้อมูลในรูปแบบไบต์ช่วยให้การบริหารจัดการข้อมูลมีความสะดวกสบายและรวดเร็ว เช่น

การตรวจสอบความซ้ำซ้อน ระบบจะทำการตรวจสอบว่ามีข้อมูลครูที่ซ้ำซ้อนอยู่ในไฟล์หรือไม่ก่อนที่จะทำการบันทึกข้อมูลใหม่ หากมีการพยายามเพิ่มข้อมูลที่มีรหัสหรือชื่อซ้ำ จะมีการแสดงข้อความข้อผิดพลาดและไม่อนุญาตให้บันทึกข้อมูลนั้น

การอัปเดตข้อมูล เมื่อมีการเปลี่ยนแปลงข้อมูลครู เช่น การเปลี่ยนชื่อหรือรายละเอียดการติดต่อ ระบบสามารถทำการอัปเดตข้อมูลที่มีอยู่ในไฟล์ teacher.bin ได้อย่างมีประสิทธิภาพ โดยใช้ฟังก์ชัน update_teacher ซึ่งจะทำการเขียนข้อมูลใหม่ทับข้อมูลเดิม

การเชื่อมโยงข้อมูล ระบบจะมีการลบข้อมูลนักเรียนที่มีครูที่ถูกลบออกจากระบบ ซึ่งช่วยให้ข้อมูลในฐานข้อมูลมีความถูกต้องและเป็นปัจจุบัน

3.8 ไฟล์ report.txt

ไฟล์ report.txt มีบทบาทสำคัญในการบันทึกข้อมูลเกี่ยวกับการดำเนินการในระบบ เช่น รายงานการเพิ่ม ลบ หรืออัปเดตข้อมูลนักเรียนและครู การจัดเก็บข้อมูลในไฟล์นี้ช่วยให้ผู้ดูแลระบบสามารถติดตามการเปลี่ยนแปลงในข้อมูลได้อย่างมีประสิทธิภาพ

ข้อมูลที่สามารถบันทึกลงในไฟล์นี้อาจรวมถึง

วันที่และเวลา แสดงถึงช่วงเวลาที่มีการดำเนินการ เช่น การเพิ่มข้อมูลครูหรือนักเรียน

รายละเอียดการดำเนินการ ข้อมูลเกี่ยวกับการดำเนินการที่เกิดขึ้น เช่น "เพิ่มครูใหม่ชื่อ John Doe" หรือ "ลบนักเรียนที่มีรหัส 12345"

ข้อผิดพลาด หากมีการดำเนินการที่ไม่สำเร็จ ระบบจะบันทึกข้อความแสดงข้อผิดพลาดลงในไฟล์นี้ เช่น "ไม่สามารถเพิ่มครูได้เนื่องจากชื่อซ้ำ"

การใช้ไฟล์ report.txt ช่วยให้ผู้ดูแลระบบสามารถ

ติดตามประวัติการดำเนินการ ทำให้สามารถตรวจสอบเหตุการณ์ที่เกิดขึ้นในระบบได้ โดยไม่จำเป็นต้องตรวจสอบโค้ดหรือข้อมูลที่ถูกจัดเก็บในไฟล์หลัก

การวิเคราะห์ข้อมูล การมีรายงานที่ชัดเจนจะช่วยให้สามารถวิเคราะห์แนวโน้มของการใช้งานระบบ และปรับปรุงให้ดียิ่งขึ้นในอนาคต

การตรวจสอบความถูกต้องของข้อมูล ผู้ดูแลระบบสามารถใช้ข้อมูลในรายงานเพื่อตรวจสอบความถูกต้องของข้อมูลที่เก็บอยู่ในฐานข้อมูล

report.txt

```
Student Survey Report
The teacher has students:
    FLUK: 1
Number of students in each year:
    Year 5: 1
Number of students in each department:
    BB: 1
Total number of students:   1
```

ภาพที่ 3-45 ตัวอย่างข้อมูลในไฟล์

3.9 ไฟล์ enum_1.py

ไฟล์ enum_1.py ช่วยในการจัดระเบียบและสร้างความเข้าใจเกี่ยวกับข้อมูลที่ใช้บ่อยในระบบการศึกษา โดยการเก็บค่าคงที่เหล่านี้ไว้ในไฟล์นี้ ทำให้โปรแกรมสามารถเข้าถึงและใช้ค่าที่กำหนดได้ง่ายดายและมีความชัดเจนมากขึ้น นอกจากนี้ยังช่วยลดความซ้ำซ้อนในการประกาศค่าคงที่ในหลาย ๆ จุดของโปรแกรม เมื่อมีการเปลี่ยนแปลงหรือเพิ่มเติมแผนกใหม่ เพียงแค่ปรับปรุงในไฟล์นี้ก็จะทำให้ทั่วทั้งโปรแกรมสามารถใช้ค่าที่อัปเดตได้โดยอัตโนมัติ

```python
# all year
YEARS = [1, 2, 3, 4]
# all department
DEPARTMENTS = ['MM', "IT", "CA", "IEM", "INE", "AFE", "ITI", "IMT", "MMT", "CDM", "AFET", "INET"]
```

ภาพที่ 3-46 ไฟล์ enum_1.py