Project Final Report

TutoReal

เว็บแอปพลิเคชันสำหรับบริการจับคู่นักเรียนและติวเตอร์

(A web-based application for student-tutor matching service)

เสนอ

รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์

โดย

ปัญญา โลกาพัฒนา 6230324721

ปิยนนท์ เจริญพูนพาณิชย์ 6230334021

ปณิธิ วนศิริกุล 6230309321

ชนัต ธนสารโสภาภรณ์ 6231309521

นภัสกร แซ่เบ๊ 6231335821

อานนท์ อ๋องสกุล 6231374221

ชวกร เพียรทำ 6232005921

นญ กังวานธีรวัฒน์ 6232018021

Database Systems Design

2110322 semester 1/2021

Department of Computer Engineering,

Faculty of Engineering, Chulalongkorn University

สารบัญ

1) ความเป็นมาของโครงการ

ในโลกที่มีความรู้เพิ่มมากขึ้นและคนส่วนใหญ่จำเป็นต้องใช้ความรู้เพิ่มมากขึ้น การถ่ายทอดความรู้และความเข้าใจจึงมีความสำคัญและความจำเป็นมากขึ้นอย่างมีนัยสำคัญ ซึ่งความรู้เหล่านั้น ไม่จำเป็นและไม่สามารถจะหาได้เพียงแต่ในระบบการศึกษาหลักเพียงอย่างเดียว การศึกษาที่แท้จริงควรจะเกิดขึ้นเพื่อเป็นกลไกที่ขับเคลื่อนอนาคตของทุกๆคน คนทุกๆคนสมควรที่จะได้รับความรู้ที่ตนเองต้องการ ไม่ว่าจะจากทางระบบการศึกษาที่มีให้ หรือจากวิธีการและการศึกษาที่อยู่นอกเหนือจากกรอบระบบการศึกษา อย่างไรก็ตาม ปัญหาที่ยังคงอยู่ก็คือคนที่มีความรู้ความสามารถหลายๆคนไม่มีประสบการณ์ในการสอน ในขณะที่หลายๆคนก็จำเป็นที่จะต้องมีคนสอนที่มีความรู้และมีสำนวนที่สื่อสารกันได้อย่างเข้าใจ เราจึงมีความจำเป็นอย่างยิ่งที่จะสร้างแพลตฟอร์มที่เป็นพื้นที่ให้คนที่ต้องการคนสอน มาเลือกผู้สอนของตนเองที่เข้ากับสไตล์ตนเอง และเป็นสนามทดลองให้กับบุคคลที่ต้องการที่จะสอนอีกด้วย

นอกจากนี้ การสอนพิเศษ ก็มีความยากลำบากในระบบการจัดการด้วย ไม่ว่าจะเป็นเรื่องการนัดสถานที่ เรื่องการนัดวันเวลา หรือแม้กระทั่งเรื่องการจ่ายเงิน ซึ่งเมื่อการจัดการเหล่านี้ มันไม่เป็นระบบ จึงทำให้การสร้างแพลตฟอร์มที่เปิดพื้นที่ให้กับผู้ไม่มีประสบการณ์ กับ คนเรียนนั้นไม่เพียงแต่เกิดประโยชน์กับผู้สอนหน้าใหม่ และคนเรียนเท่านั้น ผู้สอนที่มีประสบการณ์ก็สามารถที่จะใช้แพลตฟอร์มนี้เพื่อใช้ประโยชน์จากระบบการจัดการที่แพลตฟอร์มนี้ได้อีกด้วย

เมื่อประมาณ สองสามปีก่อน คณะผู้จัดทำเองอยากที่จะลองสอนพิเศษดู แต่ด้วยความที่ไม่เคยมีประสบการณ์สอนคนอื่นนอกเสียจาก การสอนเพื่อนของตนเอง ซึ่งการสอนเพื่อนของคนเองนั้นง่ายกว่ามาก เพราะก็ได้เรียนมาด้วยกันมาแล้ว หนึ่งรอบ อีกทั้งผู้ฟังและผู้พูดก็ต่างรู้จักกัน การสื่อสารระหว่างการสอนจึงเป็นไปได้อย่างราบรื่น เมื่อจำเป็นจะต้องไปสอนคนอื่นที่มีพื้นฐานหรือความเข้าใจที่แตกต่างจากคณะผู้จัดทำ อีกทั้งยังไม่เคยสื่อสารกันมาก่อน การเรียนการสอนจึงเป็นไปได้ยาก การที่ต้องชำระเงินโดยที่ตัวผู้สอนนั้นไม่มีประสบการณ์และไม่มีอะไรมาการันตีผลลัพธ์ที่ได้หลังการเรียนเป็นสิ่งที่คณะผู้จัดทำไม่อยากทำ จึงเกิดความคิดของการเปิดสนามทดลองให้กับคนที่ไม่มีประสบการณ์ในการสอนได้มีโอกาสในการลองสอนดูก่อนให้เพิ่มพูนประสบการณ์ของตนเอง

แพลตฟอร์ม TutoReal จึงเป็นแพลตฟอร์ม web application ซึ่งช่วยให้นักเรียนและติวเตอร์มาพบกันในแพลตฟอร์ม เป็นสนามให้ติวเตอร์หน้าใหม่ได้ลองฝึกสอนเพื่อเป็นการเพิ่มประสบการณ์และความเชื่อมั่นในตนเอง และให้นักเรียนมีอิสระในการเลือกติวเตอร์ที่เหมาะสมให้กับตนเอง และยังเป็นระบบการจัดการการศึกษานอกห้องเรียนให้กับติวเตอร์และนักเรียนโดยทั่วๆไปอีกด้วย

2) วัตถุประสงค์ของโครงการ

- เพื่อจัดทำระบบการจับคู่นักเรียนกับติวเตอร์ที่เหมาะสมได้สะดวกจากการจัดทำฐานข้อมูลอย่างเป็นระบบ
- เพื่อพัฒนาช่องทางการเรียนพิเศษบนออนไลน์ให้สอดคล้องกับวิถีชีวิตใหม่ (new normal)
- เพื่อเพิ่มความปลอดภัยในระบบการจ่ายเงินค่าเล่าเรียนผ่านการออกแบบฐานข้อมูล
- เพื่อประเมินคุณภาพการเรียนการสอนของติวเตอร์ผ่านการออกแบบฐานข้อมูลที่ดี

3) ฟังก์ชันการทำงานของระบบ (System Functionalities)

1. ระบบลงทะเบียนและยืนยันตัวตน
   - 1.1. ผู้ใช้งานสามารถสมัครสมาชิกด้วย email ที่ไม่ซ้ำกับบัญชีอื่น
   - 1.2. ผู้ใช้งานสามารถสมัครสมาชิกด้วยรหัสผ่านที่กำหนดขึ้นมาเองได้
   - 1.3. ผู้ใช้งานสามารถเลือกประเภทบัญชีได้เพียงแบบเดียวต่อ 1 บัญชี จากประเภทบัญชีทั้ง 2 แบบ ได้แก่ นักเรียนและติวเตอร์
   - 1.4. ผู้ใช้งานสามารถแก้ไขข้อมูลส่วนตัวที่กรอกไปหลังจากสมัครได้
   - 1.5. ผู้ใช้งานสามารถเปลี่ยนรหัสผ่านได้
   - 1.6. หลังจากกรอกข้อมูลสมัครสมาชิกแล้ว ผู้ใช้งานต้องยืนยันการสมัครสมาชิกผ่านอีเมล การสมัครสมาชิกจึงจะเสร็จสมบูรณ์
   - 1.7. นักเรียนต้องยืนยันตัวตนด้วยหมายเลขโทรศัพท์ก่อน จึงจะสามารถจองเวลาเรียนกับติวเตอร์ได้
   - 1.8. ติวเตอร์ต้องยืนยันตัวตนด้วยบัตรประชาชนก่อน จึงจะสามารถเปิดรับการจองเวลาเรียนจากนักเรียนได้
   - 1.9. ติวเตอร์ต้องยืนยันวุฒิ/ความสามารถที่ตัวเองมี โดยการแนบเอกสารอื่นๆ เช่น ใบผลการศึกษา มาในระบบ เพื่อให้แอดมินตรวจสอบได้
2. ระบบเข้าสู่ระบบ (Login)
   - 2.1. ผู้ใช้งานที่มีบัญชีแล้ว สามารถเข้าสู่ระบบและออกจากระบบได้
   - 2.2. ระบบต้องตรวจสอบ username และ password ของผู้ใช้งานเมื่อผู้ใช้งานคนนั้นทำการเข้าสู่ระบบ
   - 2.3. ผู้ใช้งานสามารถแจ้งขอเปลี่ยน password ได้ ในกรณีที่ลืม password
3. ระบบค้นหาพร้อมตัวกรอง
   - 3.1. ผู้ใช้งานทั่วไปสามารถค้นหาโปรไฟล์ติวเตอร์จากชื่อหรือวิชาหรือเนื้อหาที่สอนได้
   - 3.2 ผู้ใช้งานทั่วไปสามารถดูรายละเอียดการสอนของติวเตอร์ได้จากหน้าโปรไฟล์ของติวเตอร์คนนั้น
4. ระบบจัดตารางเวลา

4.1. ติวเตอร์สามารถสร้างตารางเวลาแล้วเลือกเวลาที่ตัวเองต้องการสอนได้ และสามารถแก้ไขตารางนี้ได้ในภายหลัง

4.2. ผู้ใช้งานสามารถเรียกดูตารางเวลาของติวเตอร์แต่ละคนได้ผ่านหน้าโปรไฟล์ของติวเตอร์คนนั้นๆ

4.3. นักเรียนสามารถจองเวลาเรียนกับติวเตอร์ โดยเลือกเวลาที่ต้องการจากตารางเวลาของติวเตอร์คนนั้นๆ การจองเวลาเรียนนี้สามารถจองได้แค่แบบเดี่ยว

4.4. ตารางเวลาจะอัพเดทอัตโนมัติตามเวลาที่ถูกจอง

4.5. เมื่อจองเวลาแล้ว ระบบจะสร้าง link online meeting ที่จะใช้สอนให้นักเรียน

4.6. ผู้ใช้งานสามารถดูรายการจองเวลาเรียนหรือเวลาสอนของตนเองได้

4.7 ผู้ใช้งานสามารถดูรายละเอียดการจองเวลาเรียนหรือเวลาสอนของตนเองได้ในแต่ละอัน

4.8. นักเรียนสามารถดูตารางเวลาการเรียนที่ตัวเองจองไปแล้วได้

4.9. ผู้ใช้งานสามารถยกเลิกการจองได้ โดยต้องยกเลิกก่อนจะถึงเวลาที่นัดหมาย 1 วัน และต้องได้รับความยินยอมจากอีกฝ่ายที่มีส่วนร่วมในการจองเวลาเรียน

4.10. ผู้ใช้งานสามารถแก้ไขเวลาที่นัดหมายในการจองเวลาเรียนได้ โดยต้องแก้ไขก่อนที่จะถึงเวลาที่นัดหมาย 1 วัน และต้องได้รับความยินยอมจากอีกฝ่ายที่มีส่วนร่วมในการจองเวลาเรียน

4.11 เมื่อการจองเวลาเรียนถูกยกเลิก เงินที่นักเรียนชำระค่าเรียนมาจะถูกโอนคืนให้นักเรียน

5. ระบบแชท

5.1. นักเรียนสามารถส่งข้อความผ่านแชทหาติวเตอร์คนใดก็ได้ เพื่อสอบถามข้อมูลเกี่ยวกับการเรียน

5.2 ติวเตอร์สามารถตอบข้อความ และส่งข้อความหานักเรียนที่ส่งข้อความมาหาติวเตอร์ได้

6. ระบบชำระเงิน

6.1. นักเรียนต้องชำระเงินค่าเรียนเข้ามาในระบบหลังจากการจองเวลาเรียน

6.2. นักเรียนสามารถใช้บัตรเครดิตหรือการโอนเงินผ่านบัญชีธนาคารในการชำระเงินก็ได้

6.3. ติวเตอร์จะได้รับเงินค่าเรียนจากระบบหลังจากการสอนเสร็จสิ้น

6.4 ระบบจะใช้ตัวกลางในการชำระเงินระหว่างนักเรียนและติวเตอร์

6.5 ก่อนที่จะโอนเงินให้กับติวเตอร์ เราจะเก็บค่าธรรมเนียม 5% ของค่าเรียน

7. ระบบรีวิว

7.1. นักเรียนที่เข้าเรียนกับติวเตอร์คนใดๆ ไปแล้วอย่างน้อย 1 ครั้ง สามารถเขียนข้อเสนอแนะและคำวิจารณ์ และให้คะแนนติวเตอร์คนนั้นๆได้

7.2. ผู้ใช้งานทั่วไปสามารถมองเห็นข้อเสนอแนะ คำวิจารณ์และคะแนนเฉลี่ยของติวเตอร์ทุกคนได้ในหน้าโปรไฟล์ของติวเตอร์คนนั้นๆ โดยแต่ละรีวิวนั้นจะไม่เปิดเผยชื่อผู้เขียน

8. ระบบแนะนำติวเตอร์

8.1. ติวเตอร์สามารถจ่ายเงิน เพื่อให้ติวเตอร์คนนั้นปรากฏอยู่ในลำดับต้นๆของผลการค้นหาจากระบบค้นหา

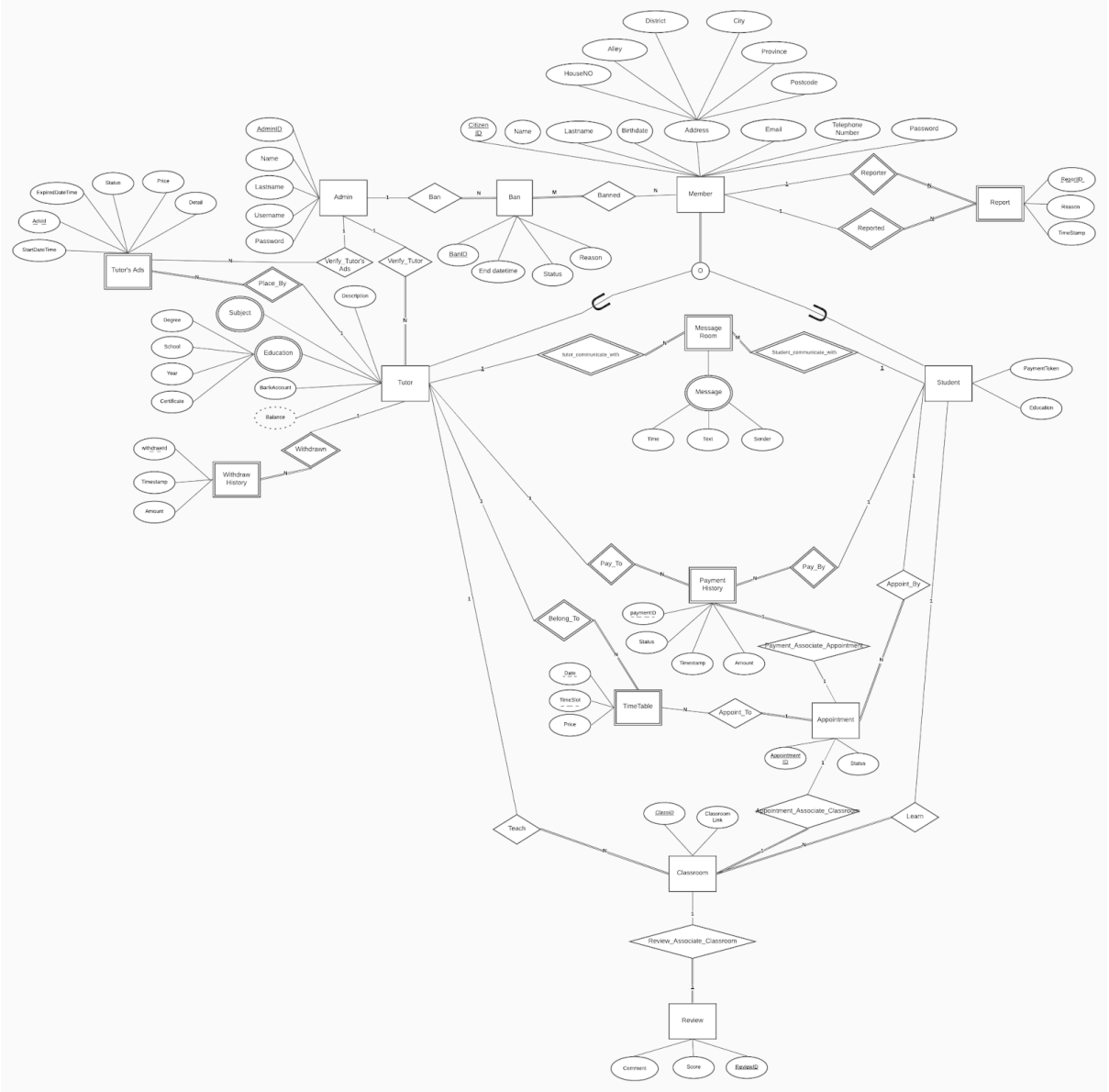
9. ระบบการรายงานและแบนผู้ใช้งาน

9.1 ผู้ใช้งานสามารถรายงานผู้ใช้งานคนอื่นๆที่มีพฤติกรรมการใช้งานที่ไม่เหมาะสมให้แอดมินทราบได้

9.2 ระบบสามารถเรียกดูรายชื่อผู้ใช้งานที่ถูกรายงานได้

9.3 แอดมินสามารถแบนผู้ใช้งานคนใดๆก็ได้ไม่ให้ใช้ระบบ โดยสามารถเลือกระยะเวลาในการแบนได้

4) ER Diagram



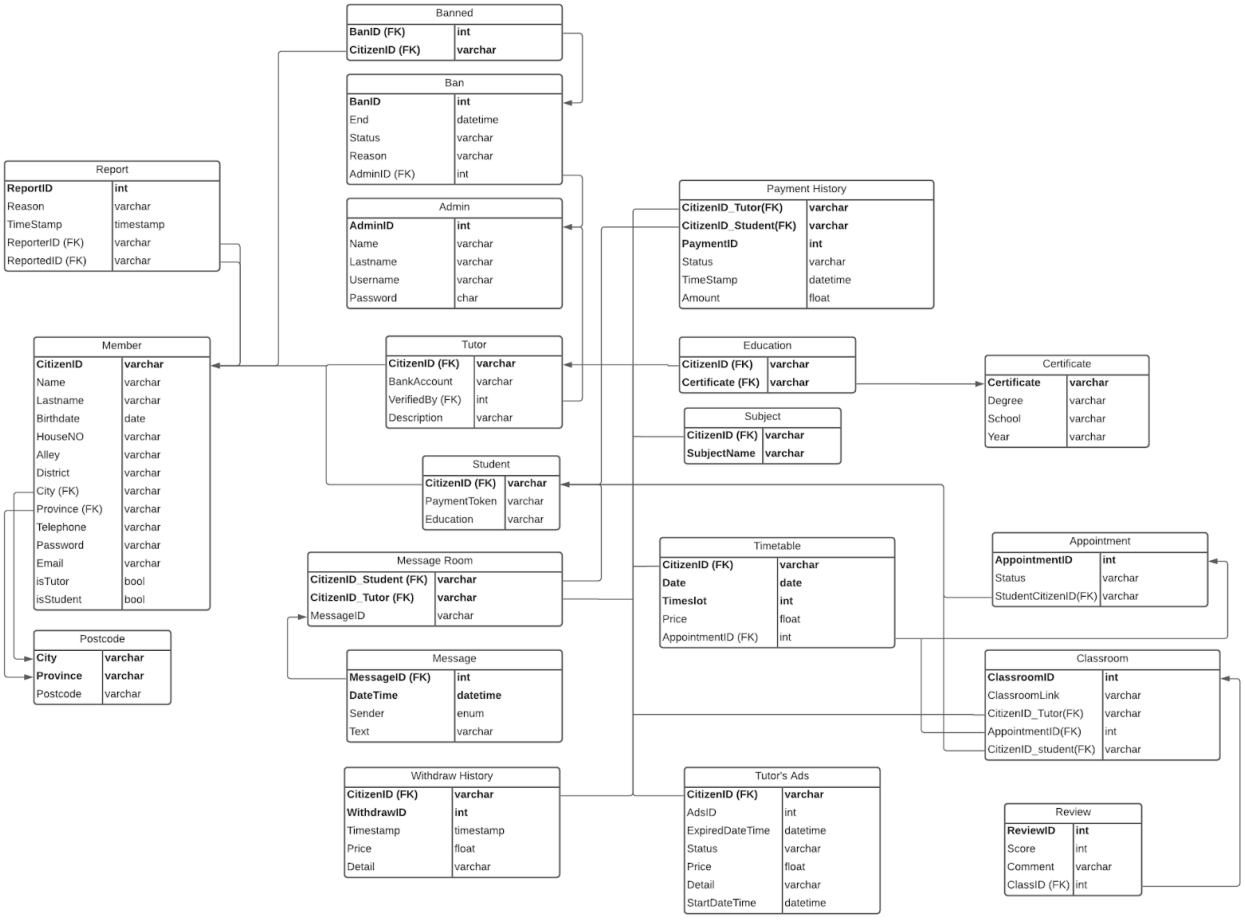
ดู ER Diagram ฉบับเต็มได้ที่ [Lucid Chart](https://lucid.app/lucidchart/6711518e-444e-4e8b-854b-c63090007edd/view?page=zbVhKap6r3FB&invitationId=inv_7362dae0-78ca-42c1-975e-fbe0f6689983#)

5) Document-based design schema

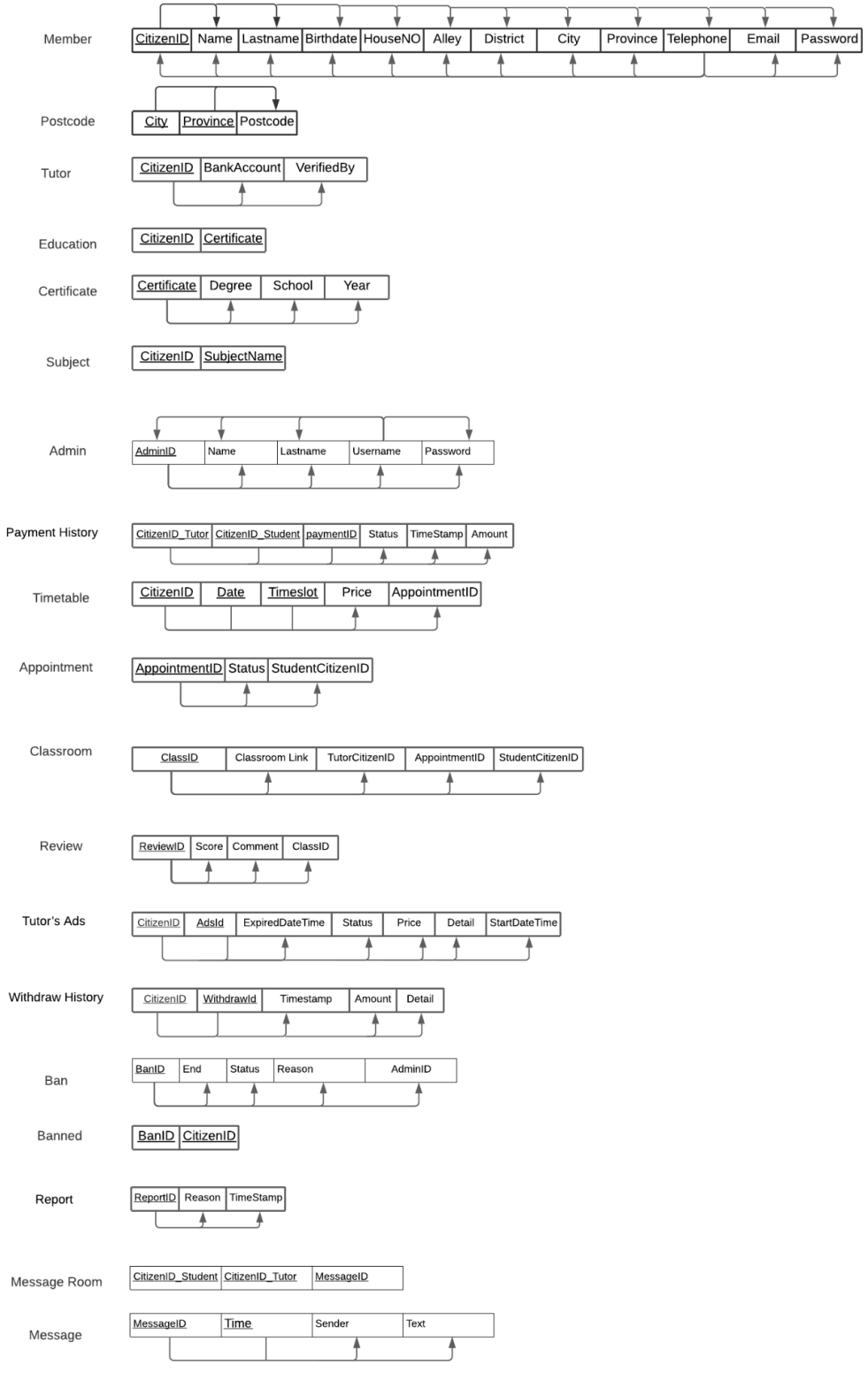
Message_Room

```
{
  bsonType: "object",
  required: [ "student_id", "tutor_id"],
  properties: {
    student_id: {
    bsonType: "objectId",
    },
    tutor_id: {
      bsonType: "string",
    },
    message: {
      bsonType: "array",
      items: {
        bsonType: "object",
        required: ["time", "text", "sender"],
        properties :{
          time: {
            bsonType: "date"
          },
          text: {
            bsonType: "string"
          },
          sender: {
            enum: ["s", "t"]
          }
        }
      }
    }
  }
}
```

6) Schema Diagram



7) Normalization



8) Data Dictionary

Relation Description

No	Relation Name	Description
1.	Member	ผู้ใช้งานที่สมัครสมาชิก
2.	Postcode	เก็บรหัสไปรษณีย์ที่สัมพันธ์กับ City, Province เพื่อเก็บข้อมูลที่อยู่
3.	Tutor	ผู้ใช้งานที่สามารถสอนในระบบได้
4.	Education	ประวัติการศึกษาของติวเตอร์
5.	Certificate	ข้อมูล/ใบรับรองการศึกษาของติวเตอร์
6.	Subject	วิชาที่ติวเตอร์ทำการเปิดสอน
7.	Student	ผู้ใช้งานที่สนใจจะเรียนกับ Tutor
8.	Admin	ผู้ดูแลระบบ
9.	Payment History	ประวัติการชำระเงินของ Tutor
10.	Time Table	ตารางเวลาของ Tutor
11.	Appointment	สัญญาการจองเวลาเรียนระหว่างติวเตอร์และนักเรียน
12.	Classroom	ห้องเรียนที่มีลิงค์ online meeting สำหรับการเรียนการสอน
13.	Review	รีวิวที่นักเรียนเขียนให้ติวเตอร์ภายในห้องเรียนที่นักเรียนได้ทำการเรียน
14.	Tutor’s Ads	โฆษณาแนะนำติวเตอร์
15.	Withdraw History	ประวัติการถอนเงินของติวเตอร์
16.	Ban	การแบนผู้ใช้งานโดยแอดมิน
17.	Banned	หมายเลขบัตรประชาชนของผู้ใช้ที่ถูกแบน
18.	Report	การรายงานผู้ใช้งานโดยผู้ใช้งาน
19.	MessageRoom	ช่องทางติดต่อสื่อสารระหว่างติวเตอร์และนักเรียน

20.	Message	ข้อความที่ถูกส่งภายใน Message Room

Relation Name: Member

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
CitizenID	VARCHAR	Citizen ID	13 Characters	No
Name	VARCHAR	Firstname	<= 100 Characters	No
Lastname	VARCHAR	Lastname	<=100 Characters	No
Birthdate	Date	Birthdate		No
HouseNO	VARCHAR	Number of the house	<= 200 Characters	Yes
Alley	VARCHAR	Name of alley	<=100 Characters	Yes
District	VARCHAR	Name of district	<=100 Characters	Yes
City	VARCHAR	Name of city	<=100 Characters	Yes
Province	VARCHAR	Name of province	<=100 Characters	Yes
Telephone	VARCHAR	Member telephone number	10 Characters	No
Password	CHAR(64)	Hashed SHA-256 member password	64 Characters	No
Email	VARCHAR	Member email	<= 45 Characters	No
isTutor	BOOL	Member as Tutor		No
isStudent	BOOL	Member as Student		No

Relation Name: Postcode

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
City	VARCHAR	City name	<= 50 Characters	No
Province	VARCHAR	Province name	<= 50 Characters	No
Postcode	VARCHAR	Postcode of the city name and province name	= 5 Characters	No

Relation Name: Tutor

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
CitizenID	VARCHAR	Citizen ID of tutor	= 13 Characters	No
BankAccount	VARCHAR	Bank Account of Tutor	10 Characters	No
VerifyBy	int	AdminID who approves Tutor		No
Description	VARCHAR	Profile description of Tutor	<=1000 Characters	Yes

Relation Name: Education

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
CitizenID	VARCHAR	Citizen ID of tutor	= 13 Characters	No
Certificate	VARCHAR	Certicate name	<= 100 Characters	No

Relation Name: Certificate

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
Certificate	VARCHAR	Certicate name	<= 100 Characters	No
Degree	VARCHAR	Degree name	<= 100 Characters	No
School	VARCHAR	School issued the degree	<= 100 Characters	No
Year	VARCHAR	Year that the degree had been issued	<= 100 Characters	No

Relation Name: Subject

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
CitizenID	VARCHAR	Citizen ID of tutor	= 13 Characters	No
SubjectName	VARCHAR	Name of a subject that the tutor wish to tutor	<= 100 Characters	No

Relation Name: Student

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
CitizenID	VARCHAR	Citizen ID of student	= 13 Characters	No
PaymentToken	VARCHAR	A token provided by the payment provider	<= 100 Characters	No
Education	VARCHAR	Education Background	<= 45 Characters	No

Relation Name: Admin

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
AdminID	INT	Admin ID	>= 0	No
Name	VARCHAR	Admin name	<= 45 Characters	No
Lastname	VARCHAR	Admin lastname	<= 45 Characters	No
Username	VARCHAR	Admin username	<= 45 Characters	No
Password	CHAR	Hashed SHA-256 admin password	<= 64 Characters	No

Relation Name: Payment History

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
CitizenID_Tutor	VARCHAR	Citizen ID of the tutor who should receive money.	= 13 Characters	No
CitizenID_Student	VARCHAR	Citizen ID of the student who paid money.	= 13 Characters	No
PaymentID	INT	Payment History ID	>= 0	No
Status	VARCHAR(10)	Payment Status	“COMPLETE”, “CANCEL” , “WAITING”	No
Timestamp	Timestamp	Payment Timestamp		No
Amount	FLOAT	Payment Amount	>= 0	No

Relation Name: Time Table

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
CitizenID	VARCHAR	Citizen ID of the tutor who is the owner of the timetable.	= 13 Characters	No
Date	DATE	date of time table		No
TimeSlot	INT	30 minutes time slot of day	0 to 47	No
Price	FLOAT	price	>= 0	No
AppointmentID	INT	Appointment ID of the appointment which related to the time table.	>=0	No

Relation Name: Appointment

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
AppointmentID	INT	Appointment ID	>=0	No
Status	VARCHAR (10)	Appointment status	“Complete”, “Cancel”, “Waiting”	No
StudentCitizenID	VARCHAR	Citizen ID of student who make this appointment	= 13 Characters	No

Relation Name: Classroom

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
ClassID	INT	Classroom ID	>=0	No
ClassroomLink	VARCHAR	Classroom link address	<= 45 Characters	Yes
CitizenID_Tutor	VARCHAR	Citizen ID of the tutor who will teach in the classroom.	= 13 Characters	No
AppointmentID	INT	Appointment ID of the appointment which related to the classroom.	>=0	No
CitizenID_Student	VARCHAR	Citizen ID of the student who will study in the classroom.	= 13 Characters	No

Relation Name: Review

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
ReviewID	INT	Review ID	>= 0	No
Score	INT	Score	0 to 10	No
Comment	VARCHAR	Comment in the Review	200 Characters	Yes
ClassID	INT	Classroom ID of the review	>= 0	No

Relation Name: Tutor’s Ads

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
CitizenID	VARCHAR	Citizen ID of tutor who owns the ads	= 13 Characters	No
AdsID	INT	Review ID	>= 0	No
ExpiredDateTime	DATETIME	Expire datetime		No
StartDateTime	DATETIME	Start datetime		No
Detail	VARCHAR	Detail of tutor’s ads	200 Characters	No
Price	FLOAT	Ads paid price	>= 0	No
Status	VARCHAR	status of ads	“WAITING”, “ACTIVE”, “EXPIRE”	No

Relation Name: Withdrawn History

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
CitizenID	VARCHAR	Citizen ID of tutor who withdraw	= 13 Characters	No
WithDrawID	INT	Review ID	>= 0	No
Timestamp	TIMESTAMP	Withdrawal timestamp		No
Amount	FLOAT	Withdrawal amount	>= 0	No
Detail	VARCHAR	Detail of withdrawn history	200 Characters	Yes

Relation Name: Ban

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
BanID	INT	Review ID	>=0	No
AdminID	INT	Admin ID	>=0	No
End Datetime	DATETIME	Due datetime		No
Status	VARCHAR	Status of the ban	“Banned”, “Dued”	No
Reason	VARCHAR	Reason of the ban	<= 100 Characters	No

Relation Name: Banned

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
BanID	INT	Ban ID	>=0	No
CitizenID	VARCHAR	Banned user ID	= 13 Characters	No

Relation Name: Report

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
ReportID	INT	Report ID	>=0	No
Timestamp	TIMESTAMP	Report timestamp		No
Reason	VARCHAR	Reason of the report	<= 45 Characters	No
RepoterID	VARCHAR	Reporter user ID	<= 45 Characters	No
ReportedID	VARCHAR	Reported user ID	<= 45 Characters	No

Relation Name: MessageRoom

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
CitizenID_Tutor	VARCHAR	ID of tutor who belong to the message room	13 Characters	No
CitizenID_Student	VARCHAR	ID of tutor who belong to the message room	13 Characters	No
MessageID	VARCHAR	Messageroom ID	<= 45 Characters	No

Relation Name: Message

Attribute Name	Data Type	Descriptive Name	Valid Values	Allow Nulls
MessageID	VARCHAR	Messageroom ID	<= 45 Characters	No
DateTime	DATETIME	The date and time that the message have been send		No
Sender	ENUM	The flag for sender identification	“S” is Student, “T” is Tutor	No
Text	VARCHAR	The text that send by sender		No

9) Indexing

เนื่องจากระบบฐานข้อมูลชุดนี้เป็นระบบฐานข้อมูลสำหรับค้นหาติวเตอร์ จึงมีความเป็นไปได้สูงที่จะมีจำนวนข้อมูลของติวเตอร์ต่าง ๆ เป็นจำนวนมาก ดังนั้น การจัดการข้อมูลของติวเตอร์แต่ละคนอย่างเป็นระบบจึงเป็นสิ่งจำเป็นในระบบฐานข้อมูลชุดนี้

จากที่กล่าวมาข้างต้น พวกเราจึงได้เลือก attribute ที่ชื่อว่า firstname และ lastname จาก Member มาเป็นตัวอย่างในการทำ Indexing เพราะว่า firstname และ lastname มีความเป็นไปได้สูงที่จะถูกใช้เป็นตัวแทนในการค้นหาติวเตอร์

ตัวอย่าง Query

```
SELECT SubjectName
FROM Member NATURAL JOIN Subject
WHERE firstname = 'fname' AND lastname = 'lname';
```

เราจะเอา firstname และ lastname ไปหา CitizenID ใน Member แล้วจึงนำ CitizenID ไปใช้ค้นหาข้อมูลที่เกี่ยวข้องกับติวเตอร์คนนั้นในตารางอื่น ๆ ต่อไป

โดยในการทำ Indexing กลุ่มเราได้เลือกใช้เป็นแบบ Unclustered Hash Indexing เพราะว่า Hash-based indexes สามารถทำ equality search ได้ดี นอกจากนี้กลุ่มเรายังได้เลือกใช้ data entry เป็นแบบ Alternative 2 เพราะว่า firstname และ lastname 1 คู่จะหมายถึง CitizenID แค่อันเดียวเท่านั้น

10) Stored procedures

1. 'update_ban_End' stored procedure มีไว้สำหรับ update วันที่จบการแบน (End) ให้เป็น end_date ของการแบนที่มี BanID เป็น ban_id

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `update_ban_End`(IN ban_id
varchar(13), IN end_date datetime)
BEGIN
    UPDATE ban
    SET End=end_date
    WHERE BanID=ban_id;
END
```

ตัวอย่างการใช้งาน

```
call update_ban_End(4, '2021-11-15 21:25:32');
```

ตัวอย่างผลลัพธ์ จะพบว่า End ของการแบนที่ BanID = 4 ได้ถูกอัพเดท

BanID	End	Status	Reason	AdminID
1	2021-11-19 21:25:32	Banned	Reason1	1
2	2021-11-19 21:25:32	Dued	Reason1	2
3	2021-11-19 21:25:32	Dued	Reason1	3
4	2021-11-15 21:25:32	Dued	Reason1	2
NULL	NULL	NULL	NULL	NULL

2. ‘update_ban_Reason’ stored procedure มีไว้สำหรับ update เหตุผลในการแบน (Reson) ให้เป็น ban_reason ของการแบนที่มี BanID เป็น ban_id

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `update_ban_Reason`(IN
ban_id varchar(13), IN ban_reason varchar(100))
BEGIN
    UPDATE ban
    SET Reason=ban_reason
    WHERE BanID=ban_id;
END
```

ตัวอย่างการใช้งาน

```
call update_ban_Reason(4, 'Cheat');
```

ตัวอย่างผลลัพธ์ จะพบว่า Reason ของการแบนที่ BanID = 4 ได้ถูกอัพเดท

BanID	End	Status	Reason	AdminID
1	2021-11-19 21:25:32	Banned	Reason1	1
2	2021-11-19 21:25:32	Dued	Reason1	2
3	2021-11-19 21:25:32	Dued	Reason1	3
4	2021-11-15 21:25:32	Dued	Cheat	2
NULL	NULL	NULL	NULL	NULL

11) Stored functions

FUNCTION TutorsAdsPriorityLevel(a_moneypaid double)
ทำหน้าที่ return priority ของ tutor ads โดยใช้จำนวนเงินที่จ่ายเป็นเกณฑ์

```
CREATE FUNCTION TutorsAdsPriorityLevel(a_moneypaid float)
RETURNS VARCHAR(10)
DETERMINISTIC
BEGIN
DECLARE lvl VARCHAR(10);
IF a_moneypaid>1000 THEN SET lvl = 'HIGH';
ELSEIF (a_moneypaid <= 1000 AND a_moneypaid>= 500) THEN SET lvl =
'MEDIUM';
ELSEIF a_moneypaid< 500 THEN SET lvl = 'LOW';
END IF;
```

```
RETURN (lvl);
END $$
```

ตัวอย่างการใช้งาน

```
SELECT * , TutorsAdsPriorityLevel(T.Price) FROM tutorads T;
```

ตัวอย่างผลลัพธ์

	CitizenID	AdsID	ExpiredDateTime	Statuss	Price	TutorsAdsPriorityLevel(T.Price)
▶	1234567891111	1	2021-07-07	WAITING	1500	HIGH
	1234567891113	2	2021-07-07	ACTIVE	200	LOW
	1234567891113	3	2021-07-07	ACTIVE	300	LOW
	1234567891114	4	2021-07-07	ACTIVE	700	MEDIUM

FUNCTION CalculateRating(rating FLOAT(3,2)) ทำหน้าที่ตอบว่า rating ที่ input เข้าไป ถือว่าเป็น rating ที่อยู่ในเกณฑ์ใด

```
DELIMITER $$
CREATE FUNCTION CalculateRating(
    rating FLOAT(3,2)
)
RETURNS VARCHAR(20)
DETERMINISTIC
BEGIN
    IF rating >= 8 THEN
        RETURN "Highly Recommended";
    ELSEIF rating>=6 THEN
        RETURN "Recommended";

    END IF;
    RETURN " ";
END$$
```

ตัวอย่างการใช้งาน

```
SELECT CalculateRating(9.55)
```

ตัวอย่างผลลัพธ์

	CalculateRating(9.55)
▶	Highly Recommended

12) Trigger

1. ‘update_status’ Trigger มีไว้สำหรับ update status ของการแบน เมื่อมีการ update ข้อมูลการแบน โดยถ้าหากยังไม่พ้นวันปลดแบน (End) status จะเป็น ‘Banned’ แต่ถ้าพ้นแล้ว status จะเปลี่ยนเป็น ‘Dued’

```
delimiter //
CREATE TRIGGER update_status BEFORE UPDATE ON ban
       FOR EACH ROW
       BEGIN
           IF NEW.End > now() THEN
               SET NEW.Status = "Banned";
           ELSE
               SET NEW.Status = "Dued";
           END IF;
       END;//
delimiter ;
```

ตัวอย่างการใช้งาน

```
call update_ban_End(4, '2021-11-28 21:25:32');
```

ตัวอย่างผลลัพธ์ จะพบว่าเมื่อ update End เป็น 2021-11-28 21:25:32 ในวันที่ 2021-11-21 trigger จะทำหน้าที่เปลี่ยน status ให้อัตโนมัติ

BanID	End	Status	Reason	AdminID
1	2021-11-19 21:25:32	Banned	Reason1	1
2	2021-11-19 21:25:32	Dued	Reason1	2
3	2021-11-19 21:25:32	Dued	Reason1	3
4	2021-11-28 21:25:32	Banned	Cheat	2
NULL	NULL	NULL	NULL	NULL

2. ‘permanent_ban_user’ Trigger มีไว้สำหรับการเก็บ Permanent_BanID และเวลาที่ user ถูกแบน

```
--CREATE TABLE permanent_ban(Permanent_BanID INT NOT NULL
AUTO_INCREMENT, Banned_Time TIMESTAMP, PRIMARY KEY(Permanent_BanID))

CREATE TRIGGER permanent_ban_user
    BEFORE DELETE ON member
    FOR EACH ROW
    INSERT INTO permanent_ban (Banned_Time)
    VALUES (NOW());
```

ตัวอย่างการใช้งาน

```
DELETE FROM member WHERE CitizenID = "1234567891112"
```

ตัวอย่างผลลัพธ์

	Permanent_BanID	Banned_Time
*	NULL	NULL

(ก่อนการ permanent_ban)

	Permanent_BanID	Banned_Time
▶	1	2021-11-21 16:13:46
*	NULL	NULL

(หลังการ permanent_ban)

13) Integrity

1. สำหรับการ insert payment history นั้น หากใส่ CitizenID ที่ไม่มีในตาราง tutor หรือ student เลย ก็จะไม่สามารถ insert record ไปได้ เนื่องจาก CitizenID_Tutor และ CitizenID_Student ของตาราง Payment History reference ไปหา CitizenID ของตาราง Tutor และ Student (reference integrity)

```
mysql> use tutoreal;
Database changed
mysql> select * from tutor;
+--------------+-------------+------------+-------------------------+
| CitizenID    | BankAccount | VerifiedBy | DescriptionTutor        |
+--------------+-------------+------------+-------------------------+
| 2222222222222 | 3333333333  |          2 | I am a military tutor. |
+--------------+-------------+------------+-------------------------+
1 row in set (0.00 sec)

mysql> select * from student;
+--------------+--------------+-----------+
| CitizenID    | PaymentToken | Education |
+--------------+--------------+-----------+
| 1111111111111 | NULL         | NULL      |
+--------------+--------------+-----------+
1 row in set (0.00 sec)
```

```
mysql> INSERT INTO paymenthistory(Citizen_Tutor,CitizenID_Student,StatusOfPayment,Amount) VALUES
    -> ('5555555555555','1111111111111','COMPLETE',2500);
ERROR 1054 (42S22): Unknown column 'Citizen_Tutor' in 'field list'
mysql> INSERT INTO paymenthistory(CitizenID_Tutor,CitizenID_Student,StatusOfPayment,Amount) VALUES

    -> ('5555555555555','1111111111111','COMPLETE',2500);
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`tutoreal`.`
paymenthistory`, CONSTRAINT `paymenthistory_ibfk_1` FOREIGN KEY (`CitizenID_Tutor`) REFERENCES `tu
tor` (`CitizenID`))
```

2. สำหรับการ ban ผู้ใช้งานแบบถาวร ระบบจะลบข้อมูลนั้นออกจาก Member เนื่องจากมีตารางที่ใช้ CitizenID ของ Member เป็น Foreign Key ข้อมูลในตารางเหล่านั้นจะต้องถูกลบออกไปด้วย จากภาพจะเห็นว่าเมื่อลบ member ที่มี CitizenID เป็น ‘1234567891114’ ออกไป ระบบจะลบข้อมูลจาก report ที่ใช้ CitizenID เป็น Foreign Key ด้วย

```
mysql> SELECT * from report;
+----------+---------+---------------------+---------------+---------------+
| ReportID | Reason  | ts                  | ReporterID    | ReportedID    |
+----------+---------+---------------------+---------------+---------------+
|        1 | Reason1 | 2021-11-21 14:54:17 | 1234567891111 | 1234567891112 |
|        2 | Reason2 | 2021-11-21 14:54:17 | 1234567891111 | 1234567891113 |
|        3 | Reason3 | 2021-11-21 14:54:17 | 1234567891113 | 1234567891114 |
|        4 | Reason4 | 2021-11-21 14:54:17 | 1234567891111 | 1234567891114 |
+----------+---------+---------------------+---------------+---------------+
4 rows in set (0.00 sec)

mysql> DELETE FROM member WHERE citizenID = '1234567891114'
    -> ;
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * from report;
+----------+---------+---------------------+---------------+---------------+
| ReportID | Reason  | ts                  | ReporterID    | ReportedID    |
+----------+---------+---------------------+---------------+---------------+
|        1 | Reason1 | 2021-11-21 14:54:17 | 1234567891111 | 1234567891112 |
|        2 | Reason2 | 2021-11-21 14:54:17 | 1234567891111 | 1234567891113 |
+----------+---------+---------------------+---------------+---------------+
2 rows in set (0.00 sec)
```

3. สำหรับการ update admin ผู้ทำการแบนแบบชั่วคราว ถ้าหากไม่มี adminID ใน table admin จะทำให้เกิด error เกี่ยวกับ Foreign Key Integrity ขึ้น อย่างเช่น ตาราง admin เป็นดังนี้

AdminID	firstname	lastname	username	password
1	aFIRSTNAME1	aLASTNAME1	admin1	password1
2	aFIRSTNAME2	aLASTNAME2	admin2	password2
3	aFIRSTNAME3	aLASTNAME3	admin3	password3
NULL	NULL	NULL	NULL	NULL

ถ้าหากต้องการ update adminID เป็น 20

```
call update_ban_AdminID(4, 20);
```

จะเกิด error ขึ้นดังด้านล่าง

Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`tutoreal2`.`ban`, CONSTRAINT `ban_ibfk_1` FOREIGN KEY (`AdminID`) REFERENCES `admin` (`AdminID`))

14) Execution path

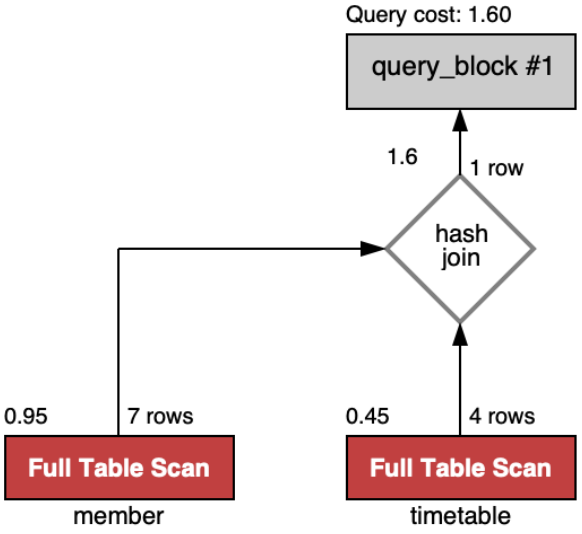
โดยนำมาจาก Usecase ที่ต้องการหาราคาต่ำสุดของค่าบริการของ Tutor โดนมีการกำหนด Firstname และ Lastname เพื่อที่จะใช้ในการ Query

คำสั่ง SQL คำสั่งที่หนึ่ง

```
SELECT MIN(price) AS price
FROM member NATURAL JOIN timetable
WHERE firstname LIKE 'FIRSTNAME1' AND lastname LIKE 'LASTNAME1';
```

คำสั่ง SQL คำสั่งที่สอง

```
SELECT MIN(t.price) AS price
FROM timetable AS t
WHERE EXISTS (
   SELECT *
   FROM member AS m
   WHERE t.CitizenID = m.CitizenID
   AND m.firstname LIKE 'FIRSTNAME1' AND m.lastname LIKE 'LASTNAME1'
);
```



15) SQL Complex query

1. SQL Complex query#1

Scenario:

ทีมของเราต้องการทำการวิเคราะห์ผลของการผลการวางโฆษณาต่อรายได้จากการสอนของติวเตอร์แต่ละคนภายในช่วงเวลาหนึ่ง โดยจะเริ่มนับจากการโอนเงินภายหลังจากที่ติวเตอร์ได้ทำการวางโฆษณาและยอดการโอนเงินนั้นจะต้องเป็นยอดที่โอนสำเร็จ โดยภายในตัวอย่างจะเริ่มนับตั้งแต่ ‘2021-09-10’ ถึง ‘2021-09-20’

```
SELECT sub1.CitizenID AS TutorID, sub1.Ads_Cost, sub2.Income
FROM (
   SELECT CitizenID, SUM(Price) AS Ads_Cost
   FROM tutor_ads
   WHERE StartDateTime BETWEEN '2021-09-10' AND '2021-09-20'
   GROUP BY CitizenID
) AS sub1
INNER JOIN (
   SELECT p.CitizenID_Tutor, SUM(amount) AS Income
   FROM payment_history AS p, (
       SELECT CitizenID, MIN(StartDateTime) AS StartDateTime
       FROM tutor_ads
       WHERE StartDateTime BETWEEN '2021-09-10' AND '2021-09-20'
       GROUP BY CitizenID
   ) AS sub2
   WHERE  p.payment_status = 'COMPLETE'
   AND (p.TimeStamp BETWEEN sub2.StartDateTime AND '2021-09-20')
   AND sub2.CitizenID = p.CitizenID_Tutor
   GROUP BY p.CitizenID_Tutor
) AS sub2 ON sub2.CitizenID_Tutor = sub1.CitizenID;
```

ตัวอย่างผลลัพธ์

```
CitizenID_tutor,Ads_Cost,Income
0000000000001,300,750
0000000000005,150,450
```

2. SQL Complex query#2

Scenario:

ทีมของเราต้องการหาชื่อของติวเตอร์ดีเด่น มาใช้ เพราะต้องการจะติดต่อกับติวเตอร์เหล่านั้นด้วยเหตุผลทางการตลาด โดยติวเตอร์ดีเด่นคือติวเตอร์ที่ไม่เคยโดน report ไม่เคยโดน cancel appointment มีจำนวน classroom ที่สอนมากที่สุด และมี rating สูงสุด โดย query จะแสดงชื่อนามสกุล และจำนวน classroom ที่สอนไป เรียงตามชื่อของติวเตอร์จากมากไปน้อย

```
    SELECT M.firstname, M.lastname, COUNT(C.CitizenID_tutor) AS
ClassCount ,AVG(RV.Score) AS AverageScore
    FROM member M
    INNER JOIN tutor T ON T.CitizenID=M.CitizenID
    INNER JOIN classroom C ON C.CitizenID_tutor=T.CitizenID
    INNER JOIN review RV ON C.ClassID=RV.ClassID
    WHERE T.CitizenID NOT IN
    (
    SELECT R.ReportedID
    FROM report R
    )
    AND T.CitizenID NOT IN
    (
    SELECT T.CitizenID
    FROM Tutor T
    INNER JOIN timetable TB ON TB.CitizenID=T.CitizenID
    INNER JOIN appointment A ON TB.AppointmentID=A.AppointmentID
    WHERE A.statuss = "CANCEL"
    )
    GROUP BY T.CitizenID
    HAVING COUNT(C.CitizenID_tutor)=(
        SELECT COUNT(C.CitizenID_tutor)
        FROM tutor T
        INNER JOIN classroom C ON C.CitizenID_tutor=T.CitizenID
        GROUP BY T.CitizenID
        ORDER BY COUNT(C.CitizenID_tutor) DESC
        LIMIT 1
    )
    AND T.CitizenID IN
    (
        SELECT CitizenID FROM
        (
        SELECT T.CitizenID , AVG(RV.Score)
        FROM Tutor T
        INNER JOIN classroom C ON C.CitizenID_tutor=T.CitizenID
        INNER JOIN review RV ON C.ClassID=RV.ClassID
        GROUP BY T.CitizenID
        ORDER BY AVG(RV.Score) DESC
        LIMIT 1
        ) a
        )
    ORDER BY M.firstname
```

ตัวอย่างผลลัพธ์

	firstname	lastname	ClassCount	AverageScore
▶	FIRSTNAME1	LASTNAME1	3	4.6667

3. SQL Complex query#3

Scenario:

ทีมของเราต้องการที่จะหาว่า นักเรียนคนไหนที่สร้างรายได้จากค่าธรรมเนียมให้เรามากที่สุดในช่วงเวลาหนึ่งๆ โดยจะนำมาทำการ track การใช้งานและความสนใจในแอพเรา และนำผลการ track ไปหา business insight ต่อสำหรับในการทำโปรโมชัน หรือปรับแผนทางการตลาดเพื่อที่จะได้สร้างลูกค้าที่สร้างรายได้สูงเพิ่มอีกได้ ซึ่งรายได้จากค่าธรรมเนียมนั้นจะแปรผันตรงกับยอดเงินที่จ่ายในแอพเรา ดังนั้นเราจึงสามารถหาว่านักเรียนคนไหนที่มียอดการจ่ายเงินรวมมากที่สุดในช่วงเวลาหนึ่งๆแทนได้

complex query สำหรับค้นว่า citizenID, firstname, lastname ของนักเรียนคนไหนที่มียอดการจ่ายเงินรวมมากที่สุดในช่วงวันที่ 19-22 พ.ย. 2021 เป็นดังนี้

```
select CitizenID_Student, firstname, lastname, sum(Amount)
from payment_history inner join member on
payment_history.CitizenID_Student = member.CitizenID
where status = 'COMPLETE' and TimeStamp between '2021-11-19' and
'2021-11-22'
group by CitizenID_Student
having sum(Amount) >= all(
    select sum(Amount)
    from payment_history
    where status = 'COMPLETE' and TimeStamp between '2021-11-19' and
'2021-11-22'
    group by CitizenID_Student
)
```

ผลลัพธ์ที่ได้

	CitizenID_Student	firstname	lastname	sum(Amount)
▶	1234567891112	FIRSTNAME2	LASTNAME2	700

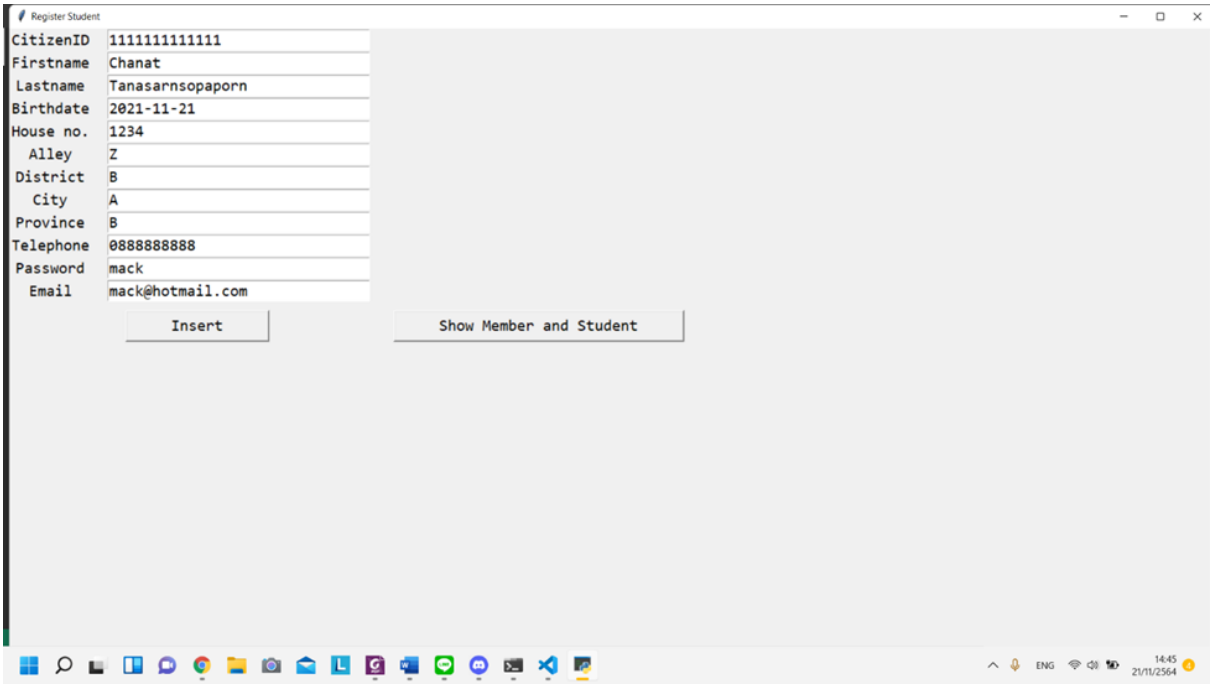
ภาคผนวก ก

UI เกี่ยวกับ INSERT member and student (Register as Student)

หน้า UI ของการ INSERT (Register as Student: หน้าการสมัครสมาชิกของผู้ใช้ที่เป็นนักเรียน)

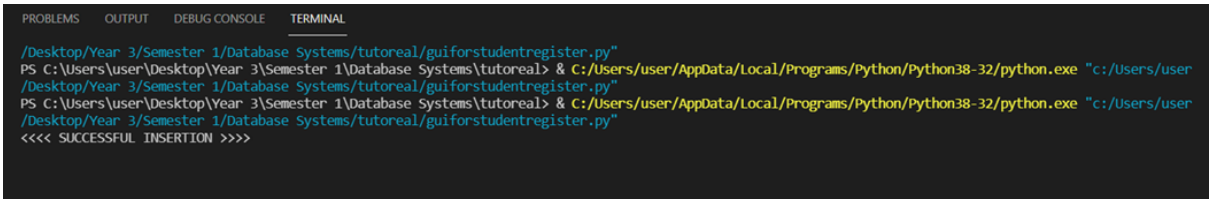
เมื่อรับข้อมูลต่างๆ ผ่านหน้า UI แล้ว ก็จะนำข้อมูลไปเพิ่ม record ทั้ง table member และ table student แต่ทว่าการ commit ข้อมูลไปสองตารางนั้นจะต้องแน่ใจว่าข้อมูลที่จะเพิ่มเข้าไปทั้งคู่นั้น valid ไม่เกิด error หรือใส่ข้อมูลผิดเงื่อนไข constraint ของทั้งสอง table จึงจะ commit ไปได้ โดยใช้การทำ TRANSACTION

1. เมื่อเรารันหน้า UI ขึ้นมาได้ ก็จะพบช่องกรอกข้อมูล เราก็กรอกข้อมูลตาม field ต่างๆ

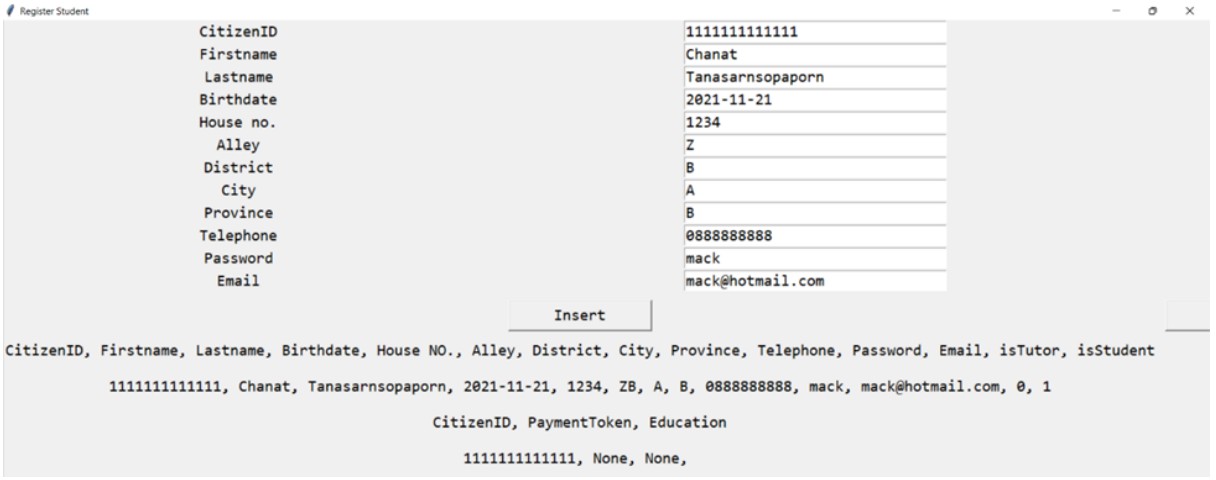


2. เมื่อกรอกข้อมูลเสร็จแล้วก็กดปุ่ม insert

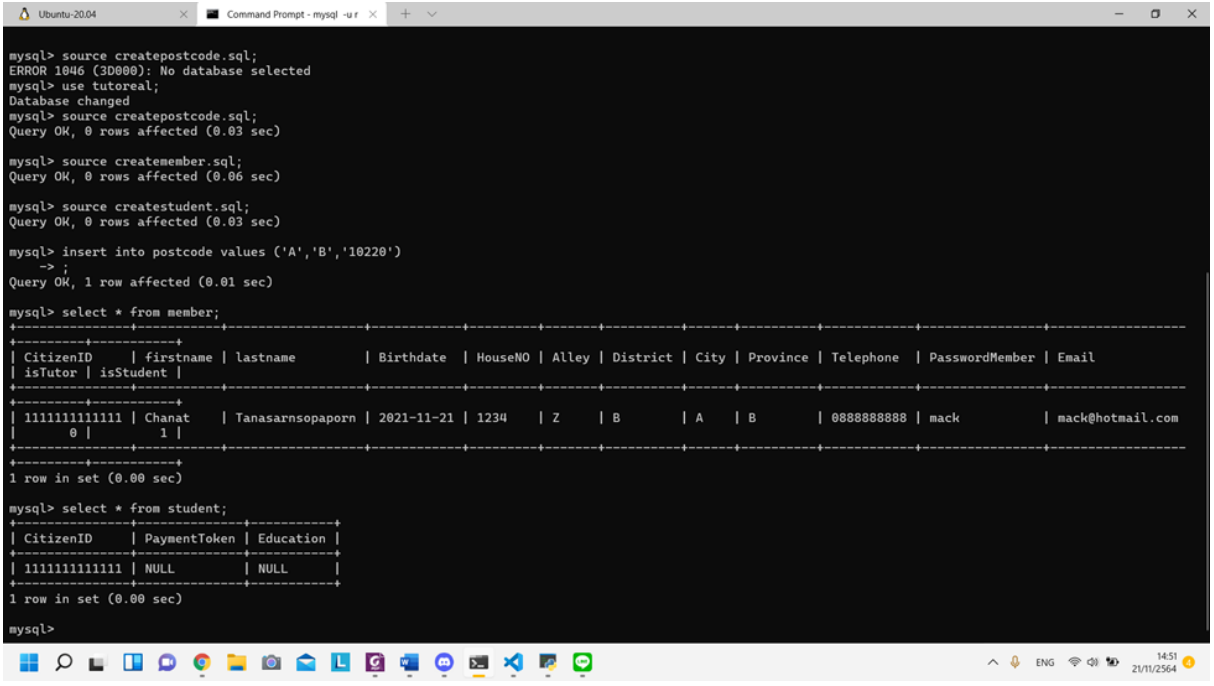
3. หาก insert สำเร็จ ข้อมูลตรง constraint ของทั้งสอง table ใน database แล้ว (member, student) หน้า Terminal ของการรัน python ก็จะ print ออกมาว่าสำเร็จ หาก commit ไม่สำเร็จ ก็จะขึ้น error จาก mysql



4. เพื่อจะเช็คว่าข้อมูลเราเพิ่มไปในตารางแล้ว ก็สามารถตรวจได้จากปุ่ม Show Member and Student จะปรากฏข้อมูลทั้งสองตาราง (ส่วนนี้ทำขึ้นเพื่อเช็คความถูกต้องของผู้จัดทำโค้ดเท่านั้น ไม่มีใน UI ของจริงแต่อย่างใด)



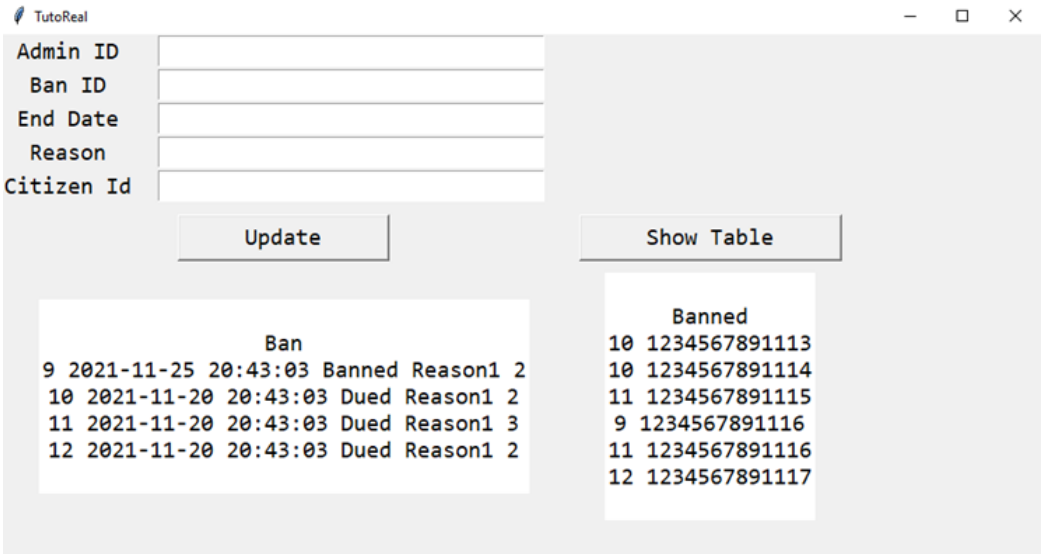
5. เพื่อเช็คว่าข้อมูลที่ใส่ไปใน UI นั้น commit จริง เราจะลอง query ข้อมูลในตารางผ่าน mysql ให้ดู



SQL for INSERT UI:

```
SET autocommit=0;

START TRANSACTION;

INSERT INTO member VALUES

('1111111111111','Chanat','Tanasarnsopaporn','2021-11-21',

'1234','Z','B','A','B','0888888888','mack','mack@hotmail.com',FALSE,
TRUE);

INSERT INTO student (CitizenID) VALUES

('1111111111111');
```

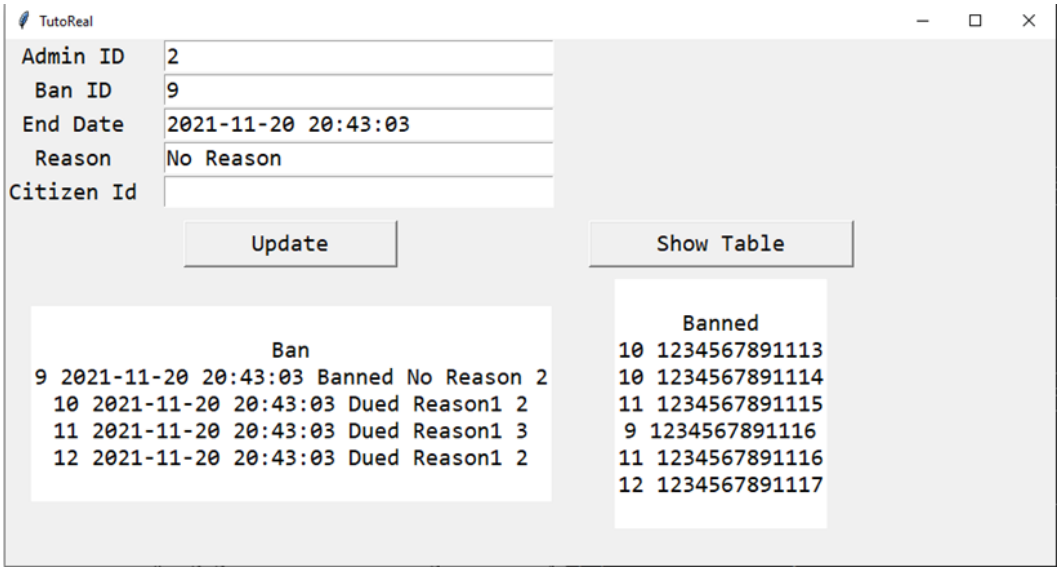
UI เกี่ยวกับ Update Ban and Banned



```
mysql> select * from ban;
+-------+---------------------+--------+---------+---------+
| BanID | End                 | Status | Reason  | AdminID |
+-------+---------------------+--------+---------+---------+
|     9 | 2021-11-25 20:43:03 | Banned | Reason1 |       2 |
|    10 | 2021-11-20 20:43:03 | Dued   | Reason1 |       2 |
|    11 | 2021-11-20 20:43:03 | Dued   | Reason1 |       3 |
|    12 | 2021-11-20 20:43:03 | Dued   | Reason1 |       2 |
+-------+---------------------+--------+---------+---------+
4 rows in set (0.00 sec)

mysql> select * from banned;
+-------+---------------+
| BanID | CitizenID     |
+-------+---------------+
|    10 | 1234567891113 |
|    10 | 1234567891114 |
|    11 | 1234567891115 |
|     9 | 1234567891116 |
|    11 | 1234567891116 |
|    12 | 1234567891117 |
+-------+---------------+
6 rows in set (0.00 sec)

mysql>
```

เมื่อต้องการอัพเดตข้อมูล ใน Table ban หรือ Table Banned ต้องต้องใส่ Admin ID และ Ban ID ลง ใน text box จากนั้นจึง สามารถอัพเดตข้อมูล End date, Reason, Citizen ID ได้ตามต้องการ เมื่อมีการกดปุ่ม Update
และ ปุ่ม Show table ไว้สำหรับ แสดงข้อมูลใน Table Ban และ Table Banned

โดยเมื่อมีการอัพเดตข้อมูล End Date หรือ Reason จะไปทำการอัพเดต ข้อมูลใน Table Ban ช่องที่มี Ban ID = Ban ID ที่กรอกลงไป



```
mysql> select * from ban;
+-------+---------------------+--------+-----------+---------+
| BanID | End                 | Status | Reason    | AdminID |
+-------+---------------------+--------+-----------+---------+
|     9 | 2021-11-20 20:43:03 | Banned | No Reason |       2 |
|    10 | 2021-11-20 20:43:03 | Dued   | Reason1   |       2 |
|    11 | 2021-11-20 20:43:03 | Dued   | Reason1   |       3 |
|    12 | 2021-11-20 20:43:03 | Dued   | Reason1   |       2 |
+-------+---------------------+--------+-----------+---------+
4 rows in set (0.00 sec)

mysql> select * from banned;
+-------+---------------+
| BanID | CitizenID     |
+-------+---------------+
|    10 | 1234567891113 |
|    10 | 1234567891114 |
|    11 | 1234567891115 |
|     9 | 1234567891116 |
|    11 | 1234567891116 |
|    12 | 1234567891117 |
+-------+---------------+
6 rows in set (0.00 sec)

mysql>
```

และเมื่อมีการอัพเดต Citizen ID จะไปการ update ข้อมูลใน Table Banned ช่องที่มี Ban ID = Ban ID ที่กรอกลงไป

```
mysql> select * from ban;
+-------+---------------------+--------+-----------+----------+
| BanID | End                 | Status | Reason    | AdminID |
+-------+---------------------+--------+-----------+----------+
|     9 | 2021-11-20 20:43:03 | Banned | No Reason |       2 |
|    10 | 2021-11-20 20:43:03 | Dued   | Reason1   |       2 |
|    11 | 2021-11-20 20:43:03 | Dued   | Reason1   |       3 |
|    12 | 2021-11-20 20:43:03 | Dued   | Reason1   |       2 |
+-------+---------------------+--------+-----------+----------+
4 rows in set (0.00 sec)

mysql> select * from banned;
+-------+---------------+
| BanID | CitizenID     |
+-------+---------------+
|    10 | 1234567891113 |
|    10 | 1234567891114 |
|    11 | 1234567891115 |
|     9 | 1234567891116 |
|    11 | 1234567891116 |
|     9 | 1234567891117 |
|    12 | 1234567891117 |
+-------+---------------+
7 rows in set (0.00 sec)

mysql>
```

Query Ban and Banned

update_ban1.sql

```
CALL start_update()
CALL update_ban_AdminID(ban_id,ban_admin)
CALL update_ban_End(ban_id,end_date)
CALL update_ban_Reason(ban_id,ban_reason)
CALL banned_del_all(ban_id)
INSERT INTO banned VALUES (BanID,CitizenID)
CALL commit_update()
```

รายละเอียดการ call ดังรูป

update_ban.sql

```
/*CALL start_update()*/
SET autocommit = 0;
START TRANSACTION;
/*CALL update_ban_AdminID(ban_id,ban_admin)*/
UPDATE ban SET AdminID=ban_admin WHERE BanID=ban_id;
/*CALL update_ban_End(ban_id,end_date)*/
UPDATE ban SET End=end_date WHERE BanID=ban_id;
/*CALL update_ban_Reason(ban_id,ban_reason)*/
UPDATE ban SET Reason=ban_reason WHERE BanID=ban_id;
/*CALL banned_del_all(ban_id)*/
DELETE FROM banned WHERE banID = ban_id;
INSERT INTO banned VALUES (BanID,CitizenID)
/*CALL commit_update()*/
COMMIT;
SET autocommit = 1;
```

UI เกี่ยวกับ DELETE report and member (ban member)

เมื่อ admin ต้องการแบนผู้ใช้ ระบบจะแสดงการรายงานทั้งหมดที่ยังไม่มีการแบน โดยจะแสดงเป็นชื่อ นามสกุล และเหตุผลของการรายงาน และด้านขวาจะมีปุ่มทั้งหมด 3 ปุ่ม โดย admin จะพิจารณาว่าควรแบนผู้ใช้คนนี้เป็นเวลาเท่าไร โดยจะเลือกได้ 3 แบบคือ 1 วัน, 1 เดือน และถาวร ดังรูป

เมื่อ admin กดปุ่มแบน 1 วันของ report ที่ 2 ระบบจะ DELETE ข้อมูล report นั้นแล้วไป INSERT ในตาราง ban และ INSERT banned ดังนั้น ui และ database จะเปลี่ยนไปดังนี้

```
mysql> SELECT * FROM banned;
+-------+--------------+
| BanID | CitizenID    |
+-------+--------------+
|     1 | 1234567891113 |
+-------+--------------+
1 row in set (0.00 sec)

mysql> SELECT * FROM ban;
+-------+---------------------+--------+---------+---------+
| BanID | End                 | Status | Reason  | AdminID |
+-------+---------------------+--------+---------+---------+
|     1 | 2021-11-22 15:38:14 | Banned | Reason2 |       1 |
+-------+---------------------+--------+---------+---------+
1 row in set (0.00 sec)

mysql> SELECT * FROM report;
+----------+---------+---------------------+---------------+---------------+
| ReportID | Reason  | ts                  | ReporterID    | ReportedID    |
+----------+---------+---------------------+---------------+---------------+
|        1 | Reason1 | 2021-11-21 15:38:14 | 1234567891111 | 1234567891112 |
|        3 | Reason3 | 2021-11-21 15:38:14 | 1234567891113 | 1234567891114 |
|        4 | Reason4 | 2021-11-21 15:38:14 | 1234567891111 | 1234567891114 |
+----------+---------+---------------------+---------------+---------------+
3 rows in set (0.00 sec)
```

เมื่อ admin กดปุ่มแบนถาวร ของ report ที่ 1 ระบบจะลบข้อมูลนั้นออกจาก Member จากรูปจะเห็นว่าไม่มี CitizenID ที่เป็น 1234567891112 ใน Member

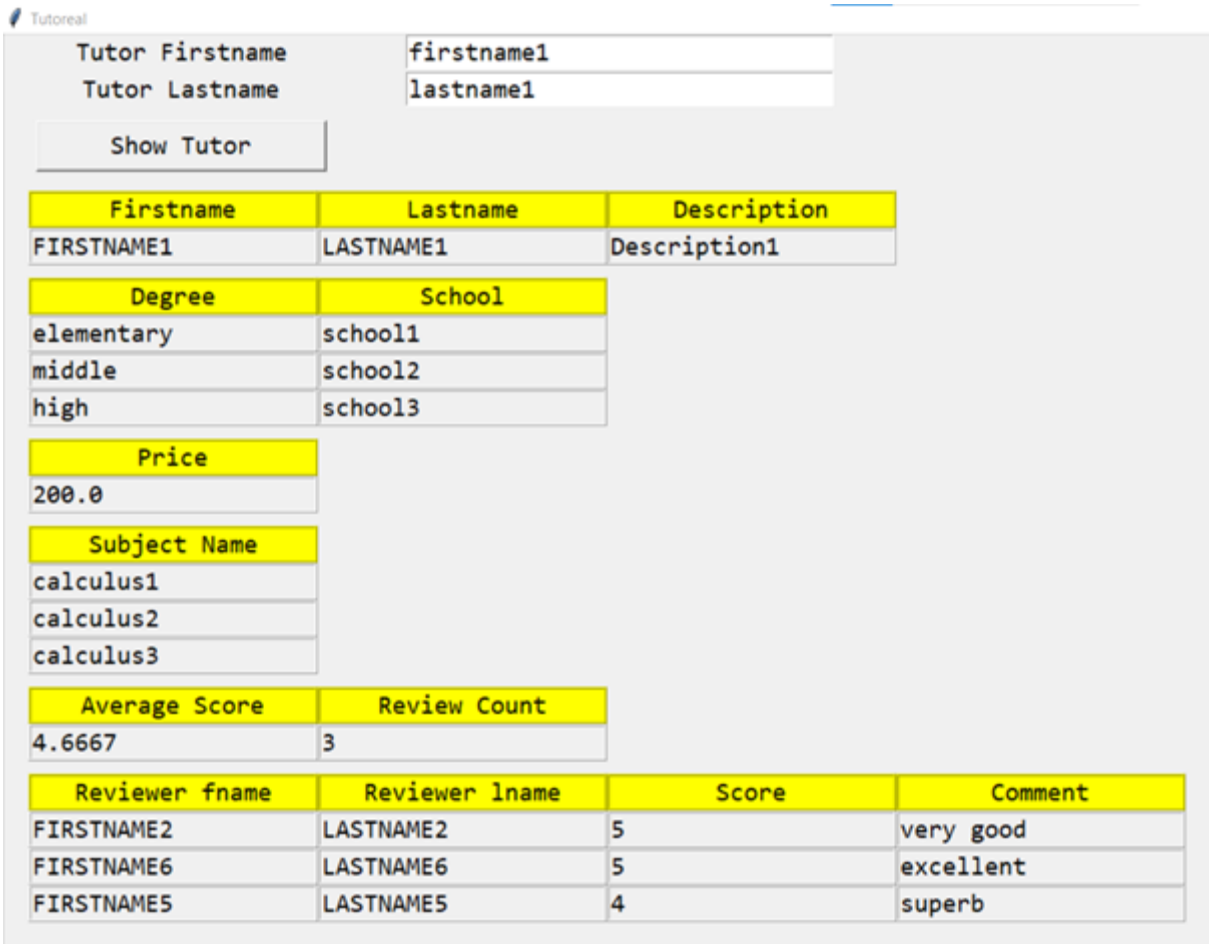
```
mysql> SELECT CitizenID FROM member;
+---------------+
| CitizenID     |
+---------------+
| 1234567891111 |
| 1234567891113 |
| 1234567891114 |
| 1234567891115 |
| 1234567891116 |
| 1234567891117 |
+---------------+
6 rows in set (0.00 sec)
```

โดยคำสั่งที่ใช้ DELETE มีดังนี้

```
my_cur.execute(f"DELETE FROM member WHERE citizenID = '{citizenID}'")
my_cur.execute(f"DELETE FROM report WHERE ReportID = {reportID};")
```

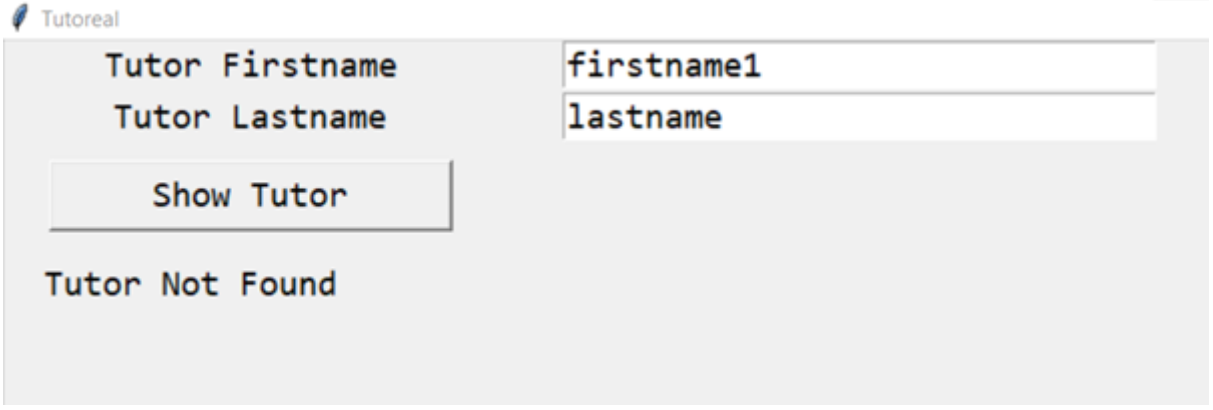
UI เกี่ยวกับ Query Profile and Review

เมื่อต้องการ Query ข้อมูลสำหรับหน้า Profile ของติวเตอร์และ Review ทั้งหมดของติวเตอร์ ให้ใส่ชื่อและนามสกุลของติวเตอร์แล้วกดปุ่ม Show Tutor



ข้อมูลที่ได้จะแยกเป็น 6 ตาราง เรียงจากบนลงล่าง ได้แก่ ชื่อของติวเตอร์, นามสกุลของติวเตอร์, คำพูดแนะนำตัว, ระดับการศึกษา, สถานศึกษา, ค่าเรียนต่ำสุด, วิชาที่เปิดสอน, คะแนนรีวิวเฉลี่ย, จำนวนรีวิวทั้งหมด, ชื่อผู้รีวิว, นามสกุลผู้รีวิว, คะแนนที่ให้, ความคิดเห็น

ถ้าหาข้อมูลติวเตอร์ไม่พบ จะแสดงคำว่า Tutor Not Found

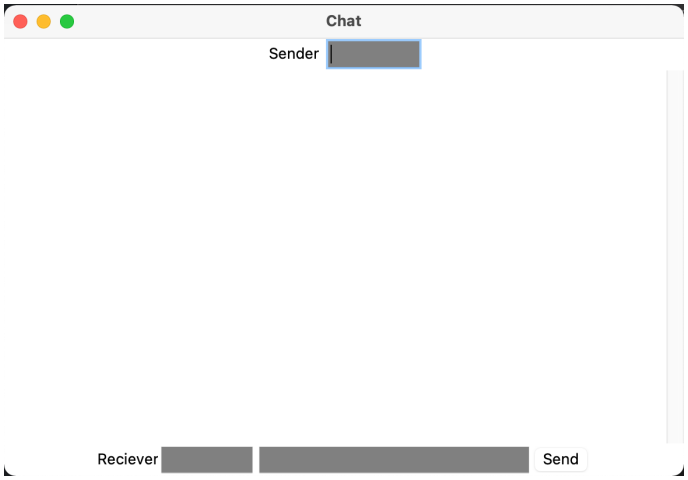


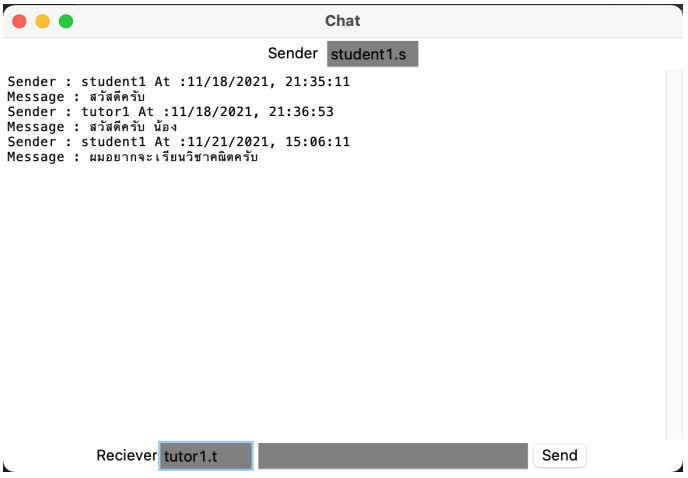
คำสั่งที่ใช้ Query ข้อมูลทั้ง 6 ตารางเรียงจากบนลงล่างเป็นดังนี้

```
command = [
    f"SELECT firstname, lastname, DescriptionTutor \
        FROM member NATURAL JOIN tutor \
        WHERE firstname LIKE '{firstname}' AND lastname LIKE '{lastname}'",

    f"SELECT Degree, School \
        FROM member NATURAL JOIN education \
        WHERE firstname LIKE '{firstname}' AND lastname LIKE '{lastname}'",

    f"SELECT MIN(price) AS price \
        FROM member NATURAL JOIN timetable \
        WHERE firstname LIKE '{firstname}' AND lastname LIKE '{lastname}'",

    f"SELECT SubjectName \
        FROM member NATURAL JOIN subjectt \
        WHERE firstname LIKE '{firstname}' AND lastname LIKE '{lastname}'",

    f"SELECT AVG(Score) AS AverageScore, COUNT(ReviewID) AS ReviewCount \
        FROM classroom NATURAL JOIN review JOIN member ON CitizenID = CitizenID_Student \
        WHERE CitizenID_Tutor = (\
            SELECT CitizenID \
            FROM member \
            WHERE firstname LIKE '{firstname}' AND lastname LIKE '{lastname}'\
        )",

    f"SELECT firstname, lastname, Score, Commentt \
        FROM classroom NATURAL JOIN review JOIN member ON CitizenID = CitizenID_Student \
        WHERE CitizenID_Tutor = (\
            SELECT CitizenID \
            FROM member \
            WHERE firstname LIKE '{firstname}' AND lastname LIKE '{lastname}'\
        ) \
        ORDER BY Score DESC",
]
```

UI เกี่ยวกับ Function ที่ใช้ NoSQL

ถูกใช้ในจัดการข้อมูลที่เกี่ยวข้องในการรับและส่งข้อความภายในระบบ โดยมีผู้ใช้งานสองฝั่ง ฝั่งละหนึ่งคน ซึ่งฝั่งหนึ่งจะเป็นนักเรียนและอีกฝั่งหนึ่งจะเป็นติวเตอร์ Schema ของข้อมูลที่ถูกนำมาใช้จะเป็น Message Room จะมี Primary key เป็น ID ของนักเรียนและติวเตอร์ที่เป็นผู้ใช้ Message Room ดังกล่าวและ ID ของ Message Room เอง และมี Attribute ประเภท Multivalue เป็น Message โดยวิธีการเก็บใน NoSQL จะ Nested เข้าไปกับ Message Room ที่จะเก็บข้อมูลเกี่ยวกับข้อความที่ใช้ในการสื่อสาร วันเวลาที่ข้อความถูกส่ง และ Flag ที่ใช้แสดงถึงผู้ที่ทำการส่ง





โดย student1.s แทน Primary key ของนักเรียนที่เป็นผู้ส่ง และ tutor1.t แทน Primary key ของติวเตอร์ที่เป็นผู้ส่ง

คำสั่งที่ใช้ในการ Insert ข้อความลงไปใน Chatroom

```
query = {
   "tutor_id": tutor,
   "student_id": student
}
message  = {
   "time" : datetime.now(),
   "text" : text,
   "sender" : sender
}
message_room_col.update_one(query, {"$push":{"message": message}})
```

คำสั่งที่ใช้ในการสร้าง Chatroom

```
chatroom = {
   "tutor_id": tutor,
   "student_id": student
}
message_room_col.insert_one(chatroom)
```

คำสั่งที่ใช้ในการหา Chatroom

```
query = {
   "tutor_id": tutor,
   "student_id": student
}
doc = message_room_col.find_one(query)
```