การจัดเก็บข้อมูลผลการเรียนผ่านสมาร์ทคอนแทร็ก บนเครือข่ายอีเธอร์เลียมเฉพาะ

DATA COLLECTION OF GRADES THROUGH SMART CONTRACTS ON ETHEREUM PRIVATE NETWORK

กิตตินันท์ อุ่นลุม
สิรินภา สามพ่วงบุญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

การจัดเก็บข้อมูลผลการเรียนผ่านสมาร์ทคอนแทร็ก บนเครือข่ายอีเธอร์เลียมเฉพาะ

DATA COLLECTION OF GRADES THROUGH SMART CONTRACTS ON ETHEREUM PRIVATE NETWORK

กิตตินันท์ อุ่นลุม
สิรินภา สามพ่วงบุญ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2561

ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การจัดเก็บข้อมูลผลการเรียนผ่านสมาร์ทคอนแทร็กบนเครือข่ายอีเธอร์เลียมเฉพาะ

DATA COLLECTION OF GRADES THROUGH SMART CONTRACTS ON ETHEREUM PRIVATE NETWORK

ผู้จัดทำ

1. นายกิตตินันท์ อุ่นลุม รหัสนักศึกษา 58010101

2. นางสาวสิรินภา สามพ่วงบุญ รหัสนักศึกษา 58011320

_____ อาจารย์ที่ปรึกษา

(อาจารย์เกียรติณรงค์ ทองประเสริฐ)

_____ อาจารย์ที่ปรึกษาร่วม

(ดร.อำนาจ ขาวเน)

การจัดเก็บข้อมูลผลการเรียนผ่านสมาร์ทคอนแทร็ก บนเครือข่ายอีเธอร์เลียมเฉพาะ

นายกิตตินันท์	อุ่นลุม	58010101
นางสาวสิรินภา	สามพ่วงบุญ	58011320
อาจารย์เกียรติณรงค์	ทองประเสริฐ	อาจารย์ที่ปรึกษา
ดร.อำนาจ	ขาวเน	อาจารย์ที่ปรึกษาร่วม

ปีการศึกษา 2561

บทคัดย่อ

ปริญญานิพนธ์นี้จัดทำขึ้นเพื่อศึกษาออกแบบและพัฒนาชิ้นงานบนระบบเครือข่ายบล็อกเชน โดยใช้สมาร์ทคอนแทร็กเพื่อทำงานบนระบบเครือข่าย เพื่อศึกษาเทคโนโลยีบล็อกเชน และเพื่อนำเทคโนโลยีนี้มาประยุกต์เพื่อให้ใช้จริงในปัจจุบัน โดยจะนำมาใช้กับการจัดเก็บข้อมูลผลการเรียนนักศึกษาในสถาบันการศึกษา ซึ่งเป็นระบบที่ต้องการความโปร่งใส ความถูกต้องของข้อมูล และเชื่อถือได้ และระบบเครือข่ายบล็อกเชนก็เป็นระบบที่ถูกออกแบบมาเพื่อรองรับความต้องการนี้ ซึ่งในปริญญานิพนธ์นี้สามารถนำข้อมูลผลการเรียนของนักศึกษามาเก็บไว้ในระบบเครือข่ายบล็อกเชน และสามารถดึงข้อมูลไปแสดงผลได้โดยทำงานผ่านทางเว็บ

Data Collection of Grades Through Smart Contracts on Ethereum Private Network

Mr. Kittinan	Ounlum	58010101
Ms. Sirinapha	Samphuangbun	58011320
Mr.Kiatnarong	Tongprasert	Advisor
Dr.Amnach	Khawne	Co-Advisor

Academic Year 2018

ABSTRACT

The purpose of this project is to study blockchain technology, to design and develop a blockchain application using smart contracts to be run on the blockchain network, to study blockchain technology, and to adapt blockchain technology to general application softwares at the present. In this project, the blockchain technology will be used to make an academic grading system in an academy. So, this system needs transparency, data correctness, and reliable. Therefore, blockchain was create for this kind of system. The system in this project can insert student scores to the blockchain and can retrieve the data to display through web application.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จลุล่วงได้ด้วยความช่วยเหลือจากหลายฝ่ายทั้งในทางตรงและทางอ้อม ปริญญานิพนธ์ฉบับนี้จะสำเร็จลงไม่ได้หากปราศจากความช่วยเหลือของบุคคลเหล่านี้ ขอขอบคุณ อาจารย์ที่ปรึกษา อาจารย์เกียรติณรงค์ ทองประเสริฐ และอาจารย์ที่ปรึกษาร่วม ดร. อำนาจ ขาวเน ที่เป็นผู้ให้คำแนะนำ คำปรึกษา และให้ความช่วยเหลือตลอดการทำปริญญานิพนธ์ ซึ่งทำให้การทำงานต่างๆ เป็นไปได้อย่างราบรื่น

ขอขอบคุณอาจารย์และบุคลากรต่าง ๆ ในสาขาวิชาวิศวกรรมคอมพิวเตอร์ที่ได้ให้คำแนะนำและสั่งสอนความรู้ต่าง ๆ มาโดยตลอด

ขอขอบคุณรุ่นพี่และเพื่อนหลาย ๆ คน ในภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้ให้คำแนะนำและคำปรึกษาอย่างเต็มที่

สุดท้ายนี้ ขอขอบคุณ บิดา มารดา และครอบครัวที่ได้เลี้ยงดู สั่งสอน และให้การสนับสนุน พร้อมทั้งให้โอกาสในการศึกษาและให้กำลังในเสมอมา

กิตตินันท์ อุ่นลุม

สิรินภา สามพ่วงบุญ

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

ตาราง **หน้า**

สารบัญรูป

บทที่ 1
บทนำ

1.1 ความเป็นมาของปัญหา

หากพูดถึงกระแสของเทคโนโลยีในเวลานี้ที่มีหลากหลายสิ่ง คงปฏิเสธไม่ได้ว่า สกุลเงินดิจิทัล (Cryptocurrency) เป็นหนึ่งสิ่งที่ได้รับความสนใจอย่างล้นหลาม ถูกคิดค้นขึ้นในปี พ.ศ.2551 ในปัจจุบันก็มีหลากหลายสกุลเงิน ไม่ว่าจะเป็น Bitcoin, Ripple, Cardano และสกุลเงินอื่น ๆ อีกกว่า 2000 สกุล โดยโครงสร้างพื้นฐานที่ใช้ก็คือ บล็อกเชน (Blockchain) ทำให้คนส่วนใหญ่มองว่า บล็อกเชน เป็นเทคโนโลยีที่สร้างมาเพื่อธุรกรรมทางด้านการเงินเท่านั้น แต่ความจริงแล้ว บล็อกเชน ยังสามารถประยุกต์ใช้กับเทคโนโลยีดิจิทัลในด้านอื่น ๆ ได้หลากหลายรูปแบบ จนมีคนบางกลุ่มกล่าวว่า เทคโนโลยีบล็อกเชน จะสามารถเปลี่ยนแปลงการดำเนินชีวิตของเราได้ในอนาคต

เทคโนโลยีบล็อกเชน จะทำหน้าที่เสมือนฐานข้อมูลหรือบัญชี (Ledger) ที่กระจายตามเครื่องคอมพิวเตอร์ต่าง ๆ ทำให้ไม่จำเป็นต้องใช้ศูนย์ข้อมูลกลางในการจัดเก็บข้อมูล นำมาซึ่งความปลอดภัย และน่าเชื่อถือ เนื่องจากไม่ต้องพึ่งพิงบุคคลที่สาม มาคอยตรวจสอบความน่าเชื่อถือของข้อมูล ซึ่งการใช้เทคโนโลยีบล็อกเชน จะทำให้สมาชิกทุกคนมีสิทธิ์เท่าเทียมกันในการรับส่งข้อมูล โดยระบบจะทำการเข้ารหัสข้อมูล เพื่อช่วยรักษาความปลอดภัยของข้อมูล

จากคุณสมบัติของบล็อกเชนที่กล่าวมานี้ ทำให้เกิดการประยุกต์ใช้ในหลากหลายด้าน เช่น ระบบ Authenticating Academic Certificates เพื่อยืนยันว่าบุคคลเหล่านั้นจบการศึกษาจากสถาบันนั้นจริง Blockchain Voting System ที่ทำให้ระบบการโหวตนั้นโปร่งใสมากขึ้น และระบบจัดเก็บข้อมูลการซื้อขายหลักทรัพย์ เป็นต้น ทางผู้จัดทำได้ศึกษาปัญหาต่าง ๆ ภายในสถาบันที่สามารถช่วยพัฒนาหรือเสริมประสิทธิภาพให้ดีขึ้นได้ด้วยคุณสมบัติที่ดีของบล็อกเชน จากการศึกษาทำให้ทราบว่า ในปัจจุบันการบันทึกคะแนนของอาจารย์ภายในสถาบันใช้การส่งข้อมูลเป็นไฟล์งานไปที่ศูนย์ข้อมูลของสำนักทะเบียน ซึ่งยังมีช่องโหว่ของระบบอยู่ในบางส่วน

ทางผู้จัดทำเล็งเห็นว่าหากใช้บล็อกเชนช่วยในการจัดเก็บข้อมูลผลการเรียนผ่านสมาร์ทคอนแทร็กบนเครือข่ายอีเธอร์เลียมเฉพาะจะช่วยเพิ่มประสิทธิภาพได้ในหลายด้านทั้งการที่สามารถตรวจสอบย้อนกลับถึงที่มาที่ไปของข้อมูลได้ และลดความเสี่ยงในการถูกโจรกรรมข้อมูลของระบบเดิม

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1) เพื่อศึกษาเทคโนโลยีบล็อกเชน

2) เพื่อประยุกต์ใช้เทคโนโลยีบล็อกเชนกับการจัดเก็บข้อมูลผลการเรียน
3) เพื่อพัฒนาการจัดเก็บข้อมูลผลการเรียนให้มีความน่าเชื่อถือและโปร่งใสมากขึ้น

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1) ระบบสามารถบันทึกข้อมูลได้อย่างถูกต้องตามที่กำหนดไว้
2) เข้าใจหลักการทำงานของ Blockchain
3) เข้าใจวิธีการพัฒนา Web Application โดยใช้ Vue.js อย่างละเอียด
4) เข้าใจวิธีการเชื่อมต่อ Blockchain โดยใช้ Web3.js

1.4 ขอบเขตของปริญญานิพนธ์

1) ระบบเครือข่าย Blockchainที่ใช้เป็นแพลทฟอร์ม Ethereum แบบ Private ด้วย Decentralized Applications และ Smart contract
2) ระบบสร้างขึ้นมาเพื่อใช้งานภายในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังเท่านั้น
3) เพื่อพัฒนาการจัดเก็บข้อมูลผลการเรียนให้มีความน่าเชื่อถือมากขึ้น ซึ่งระบบสามารถนำข้อมูลเพิ่มเข้าไปได้ แต่ไม่สามารถลบหรือย้ายตำแหน่งได้

1.5 ข้อจำกัดของปริญญานิพนธ์

1) รองรับเฉพาะการลงชื่อเข้าใช้ของนักศึกษาและอาจารย์ผ่าน E-mail ของสถาบันเท่านั้น

1.6 ตารางการดำเนินงาน

ตาราง 1.1 ระยะเวลาการดำเนินงาน

หัวข้อกิจกรรม	เดือน									
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค
1.ค้นหา และเสนอหัวข้อปริญญานิพนธ์ให้อาจารย์ที่ปรึกษา										
2.ศึกษาและทดลองใช้เทคโนโลยีที่เกี่ยวข้องกับปริญญานิพนธ์										

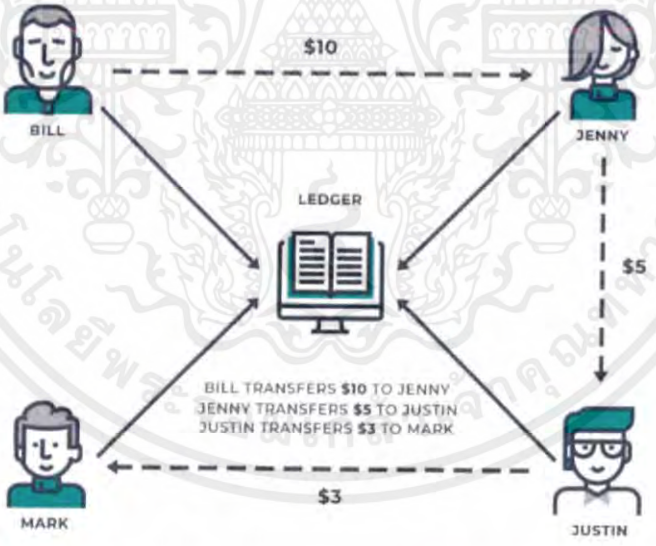
หัวข้อกิจกรรม	เดือน									
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค
3.การออกแบบ										
3.1 ออกแบบ Transaction ของ Ethereum										
3.2 ออกแบบการเชื่อมต่อ Ethereum โดยใช้ Web3.js										
3.3 ออกแบบหน้าเว็บแอป-พลิเคชันที่ใช้แสดงผล										
4.พัฒนางานโปรแกรมทั้งหมด										
5.ทดสอบและแก้ไขโปรแกรม										
6.จัดทำเอกสาร และสรุปผลปริญญานิพนธ์										

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 บล็อกเชน (Blockchain)

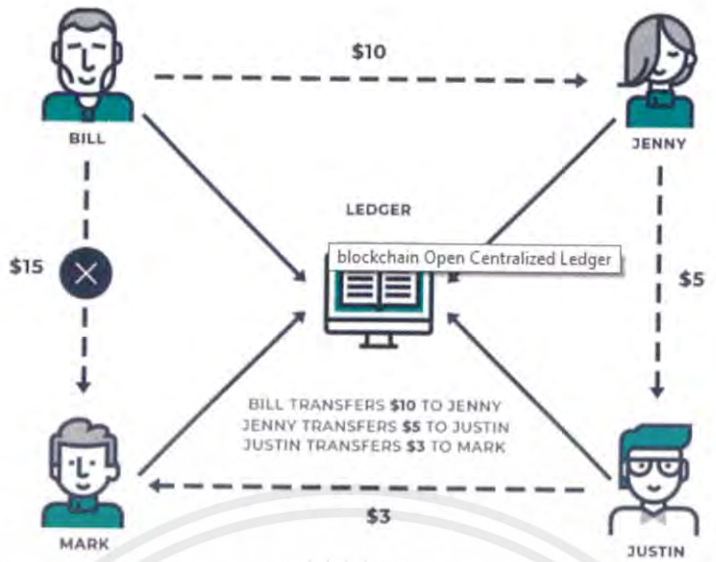
บล็อกเชน (Blockchain) คือ รูปแบบการเก็บข้อมูล (Database) แบบหนึ่ง ซึ่งมีลักษณะเป็นเครือข่าย Peer-to-Peer เก็บข้อมูลต่าง ๆ โดยไม่มีตัวกลาง ทำให้เกิดการโจรกรรมข้อมูลได้ยาก โดยหลักการของบล็อกเชน จะให้ทุกคนในเครือข่ายถือเอกสารชุดเดียวกัน เมื่อมีการเปลี่ยนแปลงก็จะเปลี่ยนแปลงด้วยกัน โดยสามารถเชื่อถือได้ว่าเอกสารเหล่านั้นเชื่อถือได้แน่นอนไม่มีการปลอมแปลง

รูปแบบการเก็บข้อมูลที่ใช้กันในปัจจุบันส่วนใหญ่ใช้เป็น ระบบรวมศูนย์กลาง (Centralized) หรือระบบที่รวมทุกสิ่งอย่างไว้ในที่ ๆ เดียว ยกตัวอย่างเช่น รูปแบบการเก็บข้อมูลของธนาคาร โดยข้อมูลจริงจะถูกเก็บไว้ที่ฐานข้อมูลของธนาคาร ส่วนสมุดบัญชีที่อยู่กับลูกค้าเป็นเพียงสำเนาที่เก็บธุรกรรมต่าง ๆ เท่านั้น



รูป 2.1 ระบบรวมศูนย์กลาง (Centralized)

ข้อดีของระบบรวมศูนย์กลาง ทำให้เกิดโอกาสผิดพลาดต่ำ เพราะมีการดูแลจากส่วนกลางเพียงอย่างเดียว ความเคลื่อนไหวทั้งหมดส่วนกลางจะรับรู้ และควบคุมได้ทั้งหมด แต่การวางระบบศูนย์กลางนี้มีค่าใช้จ่ายสูง และในแง่ความปลอดภัยต้องสามารถเชื่อใจผู้พัฒนาระบบดังกล่าว เพราะข้อมูลทุกอย่างขึ้นกับศูนย์กลางที่เดียว

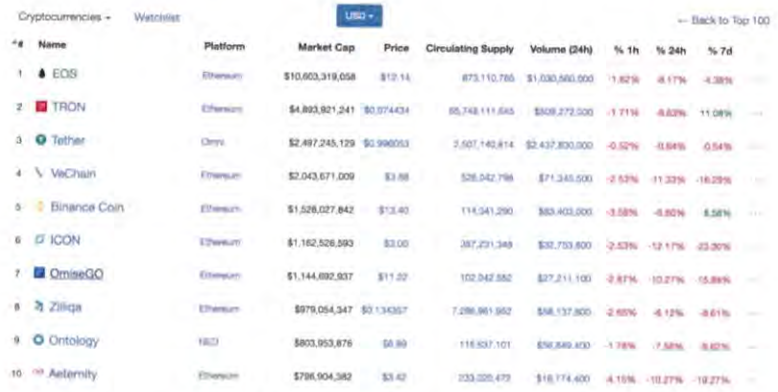


รูป 2.2 ระบบแบบกระจาย (Distributed)

บล็อกเชน จะใช้การเก็บข้อมูลแบบ ระบบแบบกระจาย (Distributed) ให้ทุกคนถือบัญชีธุรกรรม (Ledger) อิเล็กทรอนิกส์ บัญชีนี้เก็บไว้ในเครือข่ายเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ (Server) เรียกว่า โหนด (Node) แต่ละโหนดมีสำเนาบัญชีธุรกรรมของตัวเอง โดยจะเก็บรายการเดินบัญชี (Transaction) ทั้งหมดที่เกิดขึ้น การจัดการระบบดังที่กล่าวมานี้ ทำให้ธุรกรรมแต่ละรายการจะได้รับการตรวจสอบภายในเครือข่ายโดยใช้การเข้ารหัสลับที่ซับซ้อนและได้รับการตรวจสอบอย่างอิสระ ดังนั้นจึงสามารถมั่นใจในความถูกต้องของข้อมูลได้ ธุรกรรมแต่ละรายการได้รับการเก็บบันทึกอย่างต่อเนื่องและไม่มีกำหนด ดังนั้นจึงสามารถดำเนินการตรวจสอบได้ตลอดทั้งวงจรชีวิตของสินทรัพย์ สิ่งนี้จะยิ่งมีความสำคัญ อย่างมากหากข้อมูลต้นฉบับเป็นสิ่งจำเป็นที่ต้องใช้ในการตรวจสอบความถูกต้องของสินทรัพย์ และการทำธุรกรรมระหว่างบุคคลหรือกลุ่มที่เกี่ยวข้องสามารถเสร็จสมบูรณ์ได้โดยตรง และไม่ต้องมี คนกลาง ทั้งยังสามารถใช้ข้อมูลในรูปแบบดิจิทัล จึงทำให้การดำเนินธุรกรรมเป็นไปอย่างรวดเร็ว

2.2 Ethereum

Ethereum เป็นหนึ่งในสกุลเงินดิจิทัล (Cryptocurrency) ที่ใช้เทคโนโลยีบล็อกเชนทำงานอยู่เบื้องหลังและเป็นสกุลเงินที่สามารถนำไปประยุกต์ใช้ได้กับธุรกิจประเภทต่าง ๆ ได้มากมาย ไม่จำกัดเพียงการใช้ในธุรกรรมทางการเงินเท่านั้น นอกจาก Ethereum จะเป็นสกุลเงินดิจิทัลแล้ว ยังเป็นแพลตฟอร์มอีกด้วย



Cryptocurrencies ▾ Watchlist USD ▾ ← Back to Top 100

#	Name	Platform	Market Cap	Price	Circulating Supply	Volume (24h)	% 1h	% 24h	% 7d	
1	EOS	Ethereum	$10,603,319,058	$12.14	873,110,765	$1,030,560,000	-1.82%	-8.17%	-4.38%	
2	TRON	Ethereum	$4,893,921,241	$0.074434	65,748,111,645	$808,272,000	-1.71%	-8.63%	11.08%	
3	Tether	Omni	$2,497,245,129	$0.996053	2,507,140,814	$2,437,830,000	-0.52%	-0.64%	-0.54%	
4	VeChain	Ethereum	$2,043,671,009	$3.88	526,042,786	$71,345,500	-2.53%	-11.33%	-16.25%	
5	Binance Coin	Ethereum	$1,526,027,842	$13.40	114,041,290	$83,403,000	-3.58%	-6.80%	5.58%	
6	ICON	Ethereum	$1,162,526,593	$3.00	387,231,349	$32,753,800	-2.53%	-12.17%	-23.30%	
7	OmiseGO	Ethereum	$1,144,692,937	$11.22	102,042,552	$27,211,100	-2.87%	-10.27%	-15.89%	
8	Zilliqa	Ethereum	$979,054,347	$0.134357	7,286,961,952	$58,137,800	-2.65%	-6.12%	-9.61%	
9	Ontology	NEO	$803,953,876	$8.89	116,537,101	$56,849,400	-1.78%	-7.58%	-8.62%	
10	Aeternity	Ethereum	$796,904,382	$3.42	233,225,473	$18,174,400	-4.15%	-10.27%	-19.27%	

รูป 2.3 ตัวอย่างเหรียญบน Ethereum

Ethereum ยังเป็น Open-Source ทำงานบนบล็อกเชนที่ช่วยให้นักพัฒนาสามารถสร้างและใช้งานแอปพลิเคชันแบบกระจาย และมี Smart Contract ซึ่งกำหนดกติกาและบทลงโทษที่เกี่ยวกับข้อตกลงและบังคับใช้ในการใช้งาน Ethereum

2.2.1 Blockchain Consensus Protocol

Consensus Protocol เป็นส่วนสำคัญและปฏิวัติเทคโนโลยีบล็อกเชน โดย Protocol เหล่านี้สร้างระบบที่หักล้างไม่ได้ (Irrefutable System) ของข้อตกลงระหว่างอุปกรณ์ต่าง ๆ ในเครือข่ายแบบกระจาย เครือข่ายที่แข็งแกร่งนี้ปราศจากการปลอมแปลงและสร้างความไว้วางใจจากผู้มีส่วนได้เสียมากขึ้น

Blockchain Consensus Protocol เป็นสิ่งที่ทำให้โหนดทั้งหมดบนเครือข่ายมีการทำงานที่ประสานกัน ทุกโหนดในเครือข่ายจะมีสำเนาของบล็อกเชน ดังนั้น เพื่อให้แน่ใจว่าโครงสร้างข้อมูลจะถูกป้องกันการปลอมแปลง มีความปลอดภัย และถูกต้อง Consensus Protocol ทำให้แน่ใจว่าทุกด้านจะได้รับการดูแลก่อนที่จะเพิ่มบล็อกใหม่ในเครือข่ายบล็อกเชน

2.2.1.1 Proof of Work

Proof of Work ทำมาเพื่อให้การสร้างข้อมูลบนบล็อกเชนนั้นทำได้ยากและต้องพิสูจน์ตนเอง เช่นใช้พลังงานในการคำนวนหาค่า Hash ของ Block ซึ่งเป็นการคำนวนทางคณิตศาสตร์วนซ้ำไปซ้ำมาเรื่อย ๆ จนกว่าจะได้ค่าที่ถูกต้อง ดังนั้นคนที่สามารถยืนยันข้อมูลนั้นได้จึงต้องมีพลังประมวลผลที่สูงมากที่จะคำนวนหาค่านั้นได้ก่อนคนอื่น ซึ่งก็แลกมาด้วยเงินลงทุนพลังงานไฟฟ้าที่ใช้เป็นต้น และเมื่อคนนั้นยืนยันได้ ก็จะได้รางวัลเป็นผลตอบแทนไป

เมื่อมีเงินรางวัลก็จะเกิดการแข่งขันกันเพื่อยืนยันข้อมูล ระบบจึงดูน่าเชื่อถือขึ้น เพราะการสร้างบล็อกนั้นก็จะไม่ได้ขึ้นอยู่กับคนใดคนหนึ่ง เช่น บล็อกนี้สร้างโดย A ไม่ได้หมายความว่าบล็อกถัดไปจะถูกสร้างโดย A ทำให้ความเป็นไปได้ที่คนใดคนหนึ่งจะเป็นผู้ควบคุมระบบนั้นน้อยมาก

2.2.1.2 Proof of Authority

Proof of Authority ไม่ใช้การแก้ปัญหาทางคณิตศาสตร์ของโหนด แต่ใช้การอนุญาตให้โหนดที่มีสิทธิ์เท่านั้นที่จะสร้างบล็อกใหม่ได้ ข้อมูลส่วนนั้นจะต้องถูกลงความเห็นด้วยเสียงส่วนใหญ่ของโหนดที่มีสิทธิ์ตรวจสอบ วิธีนี้จะทำให้ง่ายต่อการดูแลและทำให้บล็อกเชนนั้นน่าเชื่อถือ

ในทางทฤษฎีแล้ว PoA ไม่ได้มีข้อเสียเปรียบ PoW ซึ่ง PoA นั้นมีความปลอดภัยมากกว่า เพราะว่าผู้ที่จะโจมตีไม่สามารถจะยกเลิก Transaction ทั้งหมดได้ และยังใช้การคำนวณเพื่อที่จะขุดบล็อกใหม่น้อยกว่ามาก มีประสิทธิ์ภาพที่ดีกว่า และเวลาในการสร้างบล็อกคงที่

2.3 51% Attack (Double Spend Attack)

ระบบบล็อกเชนนั้นปลอดภัย แต่ขั้นตอนการทำ Proof of Work ก็ยังมีช่องโหว่ให้ผู้ไม่หวังดีนั้นสามารถที่จะเข้ามาโจมตีระบบได้อยู่ดี โดยเฉพาะกรณีที่ผู้ไม่หวังดีนั้น มีกำลังในการขุดมากๆ เกินกว่าครึ่งหนึ่งของกำลังในระบบ โดยจะทำให้ผู้ไม่หวังดีอาจจะกลายเป็นคนเดียวที่ยืนยันข้อมูลในระบบได้เลย ซึ่งใน Public Blockchain นั้นไม่ใช่เรื่องดีแน่นอน หากคนนั้นนำพลังอำนาจไปใช้ในทางที่ผิด เรียกการโจมตีแบบนี้ว่า “51% Attack” ซึ่งอาจจะเกิดปัญหาตามมาได้ เช่น

2.3.1 เลือกที่จะไม่ยืนยัน Transaction ได้

ในกรณีนี้อาจจะทำให้ Transaction ไม่สามารถถูก Confirm ได้เลยในระบบ และไม่สามารถแก้ไขได้ เพราะไม่มีคนอื่นมาช่วยในการยืนยัน Transaction ที่ถูกต้อง

2.3.2 มี Double Spent

มีการนำ Transaction เก่าๆที่เคยถูกใช้ไปแล้วกลับมาใช้ใหม่ได้ ทำให้เงินในระบบไม่มีความหมายไปเลย

2.4 Smart Contract

Smart Contract เป็นฟังก์ชันใน Ethereum ที่อนุญาตให้สามารถเขียนโปรแกรมลงไปในข้อมูลของ Ethereum ให้ทำงานอัตโนมัติเมื่อเจอเงื่อนไขที่กำหนด มีจุดประสงค์เพื่ออำนวยความสะดวกตรวจสอบหรือบังคับใช้การเจรจาต่อรอง ความสามารถนี้ทำให้สามารถสร้างแอปพลิเคชันต่าง ๆ ขึ้นมาบนเครือข่าย Ethereum อีกชั้นหนึ่ง Smart Contract มีข้อดีหลายประการ ดังนี้

2.4.1 เป็นอิสระ

ไม่จำเป็นต้องพึ่งพาตัวกลางอื่น ๆ เพื่อยืนยันความถูกต้อง มีการจัดการอันตรายที่เกิดขึ้นจากบุคคลที่สามเนื่องจากการดำเนินการถูกจัดการโดยอัตโนมัติโดยเครือข่ายมากกว่าหนึ่งหรือมากกว่าลำเอียงอาจบุคคลที่อาจผิดพลาด

2.4.2 เชื่อถือได้

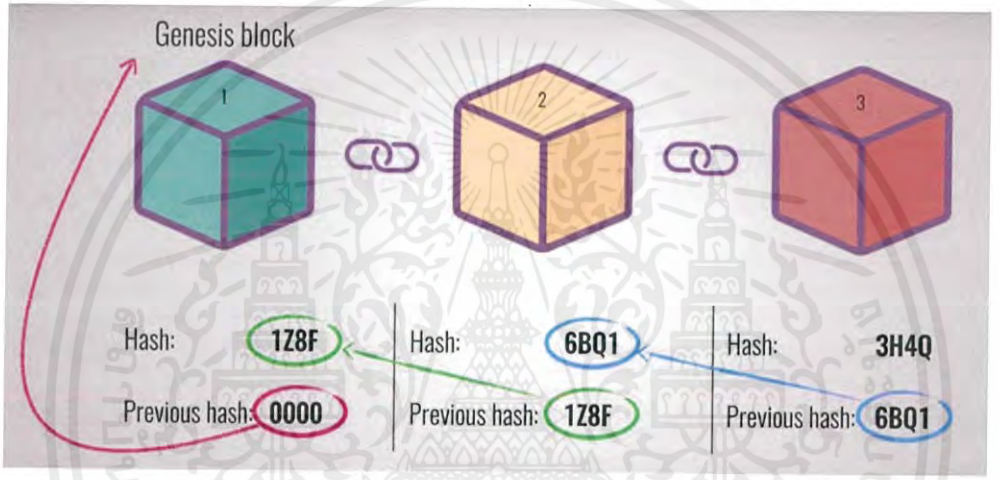
ข้อมูลถูกเข้ารหัสในบัญชีแยกประเภท (Ledger) ที่ใช้ร่วมกัน ทุกผู้ใช้ในเครือข่ายจะมีข้อมูลเดียวกัน ทำให้เชื่อถือได้ และปลอมแปลงได้ยาก

2.4.3 สำรองข้อมูล

เปรียบเทียบกับการเก็บข้อมูลไว้ที่เดียว แล้วข้อมูลเล่านั้นหายไป ก็จะไม่สามารถกู้คืนข้อมูลเล่านั้นได้เลย แต่ในบล็อกเชนผู้ใช้งานสามารถกู้คืนข้อมูลจากผู้ใช้อื่นภายในเครือข่ายได้

2.4.4 ความปลอดภัย

การเข้ารหัสลับทำให้เอกสารปลอดภัย ไม่มีการโจรกรรม



รูป 2.4 การเข้ารหัสบล็อกเชน

2.4.5 ความเร็ว

ปกติแล้วต้องใช้เวลาและเอกสารในการประมวลผลเอกสาร Smart Contract ใช้รหัสซอฟต์แวร์เพื่อทำงานโดยอัตโนมัติซึ่งจะช่วยให้ลดเวลาลง

2.4.6 ลดค่าใช้จ่าย

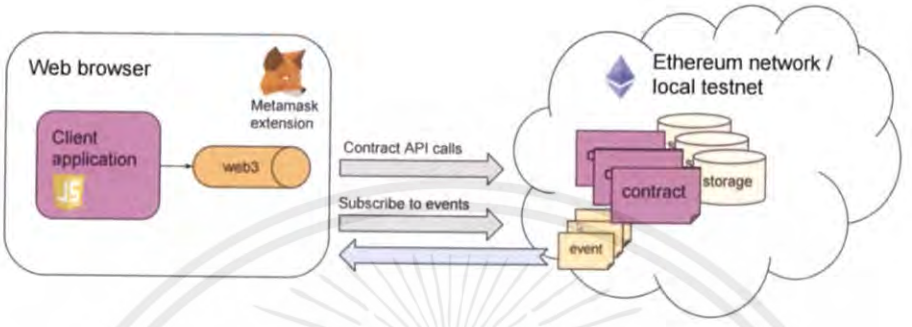
Smart Contract ช่วยให้ลดค่าใช้จ่ายในการดูแลส่วนกลาง

2.4.7 ความถูกต้อง

Smart Contract ไม่เพียง แต่ทำได้เร็วและราคาถูกเท่านั้น แต่ยังหลีกเลี่ยงข้อผิดพลาดที่เกิดขึ้นจากการผู้ใช้ได้อีกด้วย

2.5 Web3.js

Web3.js เป็น library ที่ช่วยให้สามารถโต้ตอบกับ Ethereum และสามารถอ่าน และเขียนข้อมูลจาก Smart contract ได้ สามารถสื่อสารกับ Ethereum หรือทำ Transaction กับ Smart Contact ที่ติดตั้งบนบล็อคจากภายใน JavaScript หรือเว็บแอปพลิเคชัน



รูป 2.5 การเข้ารหัสบล็อกเชน

2.6 Solidity

Solidity เป็นภาษาระดับสูงสำหรับการใช้งาน Smart Contract เป็นโปรแกรมที่ควบคุมภายใน Ethereum ซึ่งได้รับอิทธิพลจาก C ++, Python และ JavaScript และถูกออกแบบมาเพื่อกำหนดเป้าหมาย Ethereum Virtual Machine (EVM) และเป็นภาษาชนิดที่ Statically Typed และเป็นภาษาแบบ Object Oriented (OO) เพราะว่ามีคุณสมบัติของการสืบทอดและการทำ struct เป็นต้น

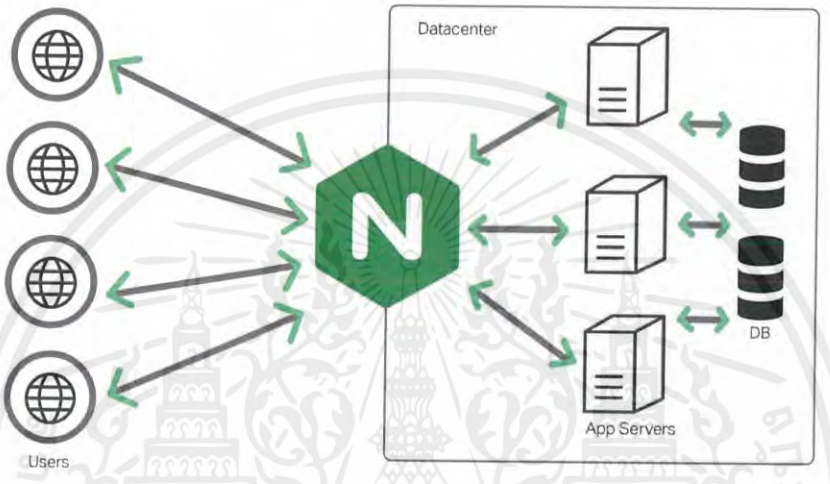
รูป 2.6 สัญลักษณ์ Solidity

2.7 NGINX

NGINX คือ ซอฟต์แวร์โอเพนซอร์สสำหรับ Web service แบบพร็อกซีย้อนกลับ การโหลดบาลานซ์สตรีมมิ่งสื่อและอื่น ๆ NGINX เป็นเว็บ Server ที่ออกแบบมาเพื่อประสิทธิภาพและความเสถียรสูงสุด นอกเหนือจากความสามารถของ Server HTTP แล้ว NGINX ยังสามารถทำหน้าที่

เป็นพร็อกซี Server สำหรับอีเมล (IMAP, POP3 และ SMTP) และพร็อกซีแบบย้อนกลับและ balancer โหลดสำหรับ Server HTTP, TCP และ UDP

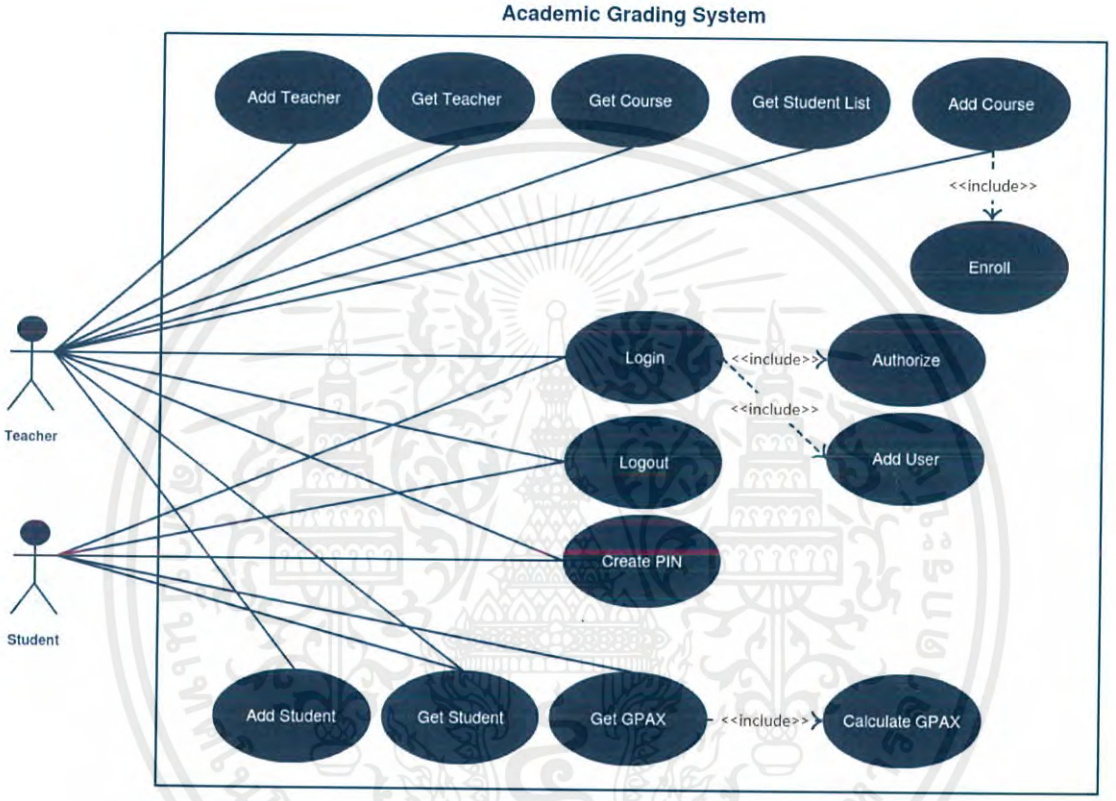
Igor Sysoev ได้เขียน NGINX เพื่อแก้ปัญหา C10K ซึ่งเป็นคำจำกัดความในปี 1999 เพื่ออธิบายความยากลำบากที่เว็บ Server ที่มีอยู่มีประสบการณ์ในการจัดการกับจำนวนมาก (10K) ของการเชื่อมต่อที่เกิดขึ้นพร้อมกัน (C) ด้วยสถาปัตยกรรมแบบอะซิงโครนัสที่อิงกับเหตุการณ์NGINX ปฏิวัติวิธีที่ Server ทำงานในบริบทที่มีประสิทธิภาพสูงและกลายเป็นเว็บ Server ที่เร็วที่สุดที่มีอยู่



รูป 2.7 การทำงานของ NGINX

บทที่ 3
การออกแบบและพัฒนา

3.1 Use Case ของระบบ



รูป 3.1 Use Case ของระบบ

ตาราง 3.1 Use case การเพิ่มผู้ใช้งานในระบบ

ID	Use case
Title	การเพิ่มผู้ใช้งานในระบบ
Description	เพิ่มผู้ใช้งานใหม่เข้าในระบบ Registration System
Actor(s)	Student, Teacher
Pre-conditions	Actor ทำการล็อคอินครั้งแรก
Post-conditions	ผู้ใช้งานถูกเพิ่มเข้าในระบบ

ID	Use case
Main success scenario	1.ผู้ใช้งานทำการล็อคอินเข้าสู่ระบบด้วย email สถาบันฯ 2.ระบบตรวจสอบว่าผู้ใช้งานอยู่ในระบบ Registration System หรือไม่ 3.ในกรณียังไม่มีผู้ใช้งานในระบบ ระบบจะทำการเพิ่มผู้ใช้งานเข้าสู่ระบบ 4.ในกรณีผู้ใช้งานยังไม่เคยกำหนด PIN ระบบจะให้ทำการกำหนด PIN เป็นตัวเลขความยาว 6 ตัว

ตาราง 3.2 Use case การเพิ่มรหัส PIN

ID	Use case
Title	การเพิ่มรหัส PIN
Description	เพิ่มรหัส PIN ให้กับผู้ใช้งานเพื่อใช้สำหรับปลดล็อคการใช้งานฟังก์ชันต่าง ๆ
Actor(s)	Student, Teacher
Pre-conditions	ผู้ใช้งานอยู่ในระบบ Registration แล้ว และยังไม่กำหนด PIN
Post-conditions	PIN ของผู้ใช้งานถูกกำหนดขึ้น
Main success scenario	1.ผู้ใช้งานทำการเรียกใช้ฟังก์ชันเช่น การเรียกดูคะแนนของ Student หรือ Teacher อัปโหลดไฟล์คะแนนขึ้นระบบ 2.ระบบตรวจสอบให้ผู้ใช้งานกรอก PIN 3.ในกรณีผู้ใช้งานยังไม่เคยกำหนด PIN ระบบจะให้ทำการกำหนด PIN เป็นตัวเลขความยาว 6 ตัว

ตาราง 3.3 Use case การเพิ่มข้อมูลคะแนนของนักศึกษาและรายวิชาในระบบ

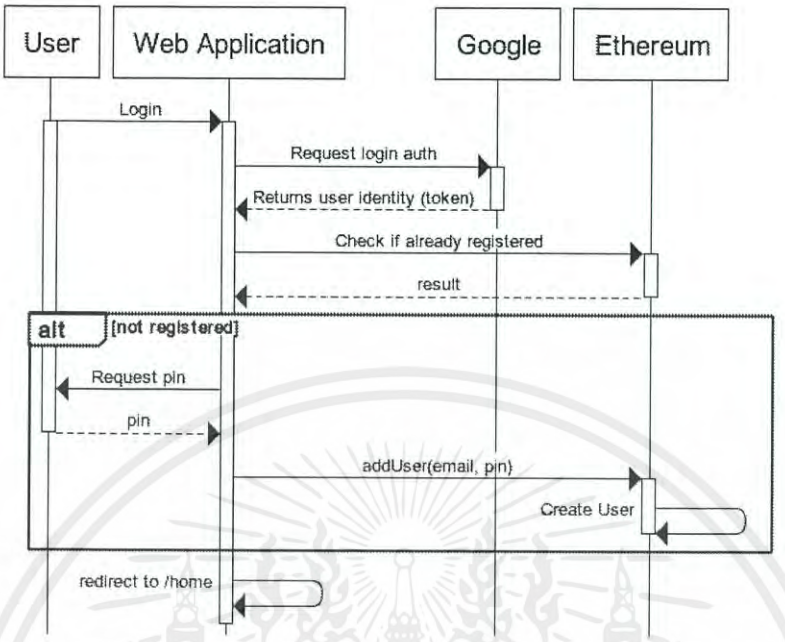
ID	Use case
Title	การเพิ่มข้อมูลคะแนนของนักศึกษาและรายวิชาในระบบ
Description	เพิ่มข้อมูลผลการเรียนของนักศึกษาเป็นรายวิชา
Actor(s)	Teacher
Pre-conditions	ผู้ใช้งานอยู่ในระบบ Registration และกำหนด PIN แล้ว
Post-conditions	ข้อมูลวิชาและคะแนนนักศึกษาถูกเก็บลงในระบบ

ID	Use case
Main success scenario	1.ผู้ใช้งานทำการเรียกใช้ฟังก์ชันการเพิ่มข้อมูลคะแนนลงไปในระบบ 3.เลือกไฟล์ที่ต้องการจะอัปโหลดโดยการคลิกที่บริเวณที่กำหนด หรือทำการลากไฟล์มาไว้บริเวณที่กำหนด 4.ผู้ใช้งานกดปุ่ม Upload เพื่ออัปโหลดข้อมูล ในกรณีที่เลือกผิดไฟล์ กดปุ่ม Cancel เพื่อยกเลิก 5.เมื่ออัปโหลดเสร็จสมบูรณ์จะขึ้นข้อความแจ้งเตือน

ตาราง 3.4 Use case การแสดงผลรายวิชา

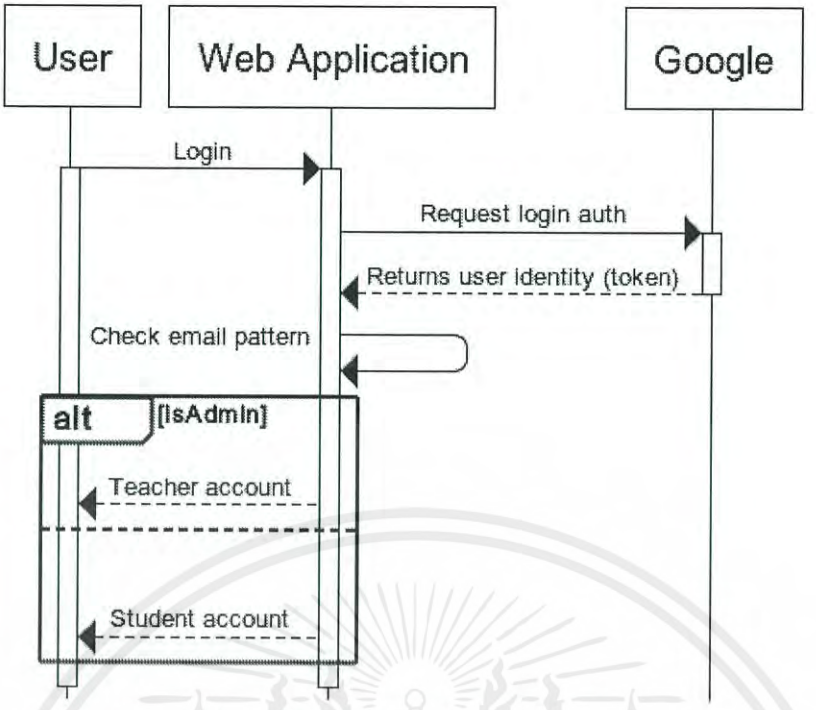
ID	UC1
Title	การแสดงผลรายวิชา
Description	แสดงผลรายวิชาที่ได้ลงทะเบียนไว้
Actor(s)	Student, Teacher
Pre-conditions	มีคะแนนรายวิชาที่ลงทะเบียนไว้แล้ว
Post-conditions	แสดงผลข้อมูลรายวิชา
Main success scenario	1.ผู้ใช้งานทำการเรียกใช้ฟังก์ชันเช่น การเรียกดูคะแนนของ Student หรือ Teacher อัปโหลดไฟล์คะแนนขึ้นระบบ 2.ระบบตรวจสอบให้ผู้ใช้งานกรอก PIN 3.เลือกไฟล์ที่ต้องการจะอัปโหลดโดยการคลิกที่บริเวณที่กำหนด หรือทำการลากไฟล์มาไว้บริเวณที่กำหนด 4.ผู้ใช้งานกดปุ่ม Upload เพื่ออัปโหลดข้อมูล ในกรณีที่เลือกผิดไฟล์ กดปุ่ม Cancel เพื่อยกเลิก 5.เมื่ออัปโหลดเสร็จสมบูรณ์จะขึ้นข้อความแจ้งเตือน

3.3 Sequence Diagram



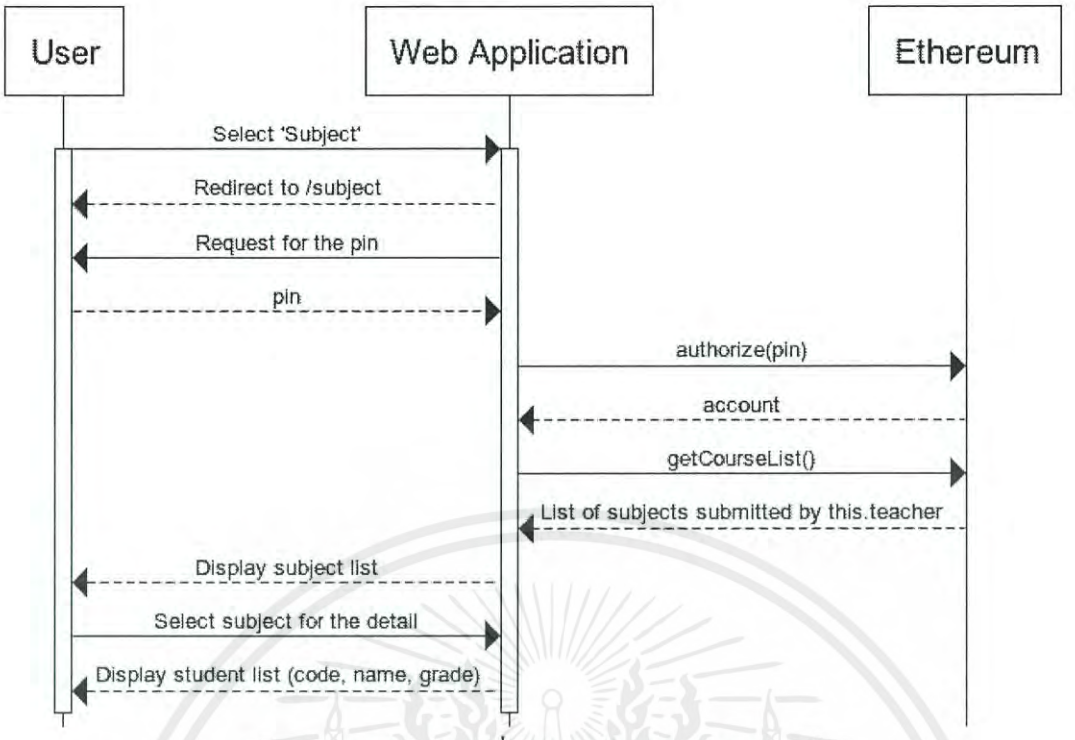
รูป 3.2 การ Login เข้าสู่ระบบ

การล็อคอินเข้าสู่ระบบเริ่มจากผู้ใช้งานเข้าระบบผ่านบราวเซอร์ บราวเซอร์จะติดต่อไปยัง Google เพื่อยืนยัน gmail ของผู้ใช้งาน เมื่อถูกต้อง จะติดต่อไปยังบล็อกเชนเซิฟเวอร์ว่ามีแอคเคาท์แอดเดรสของผู้ใช้งานนี้หรือยัง ถ้ายังจะทำการสร้างขึ้นมาใหม่และผู้ใช้งานต้องกำหนดพินขึ้นมาเพื่อใช้ปลดล็อคการทำงานส่วนอื่นๆ เมื่อตรวจสอบหรือสร้างแอคเคาท์แล้วผู้ใช้งานจึงจะเข้าสู่ระบบได้สำเร็จ



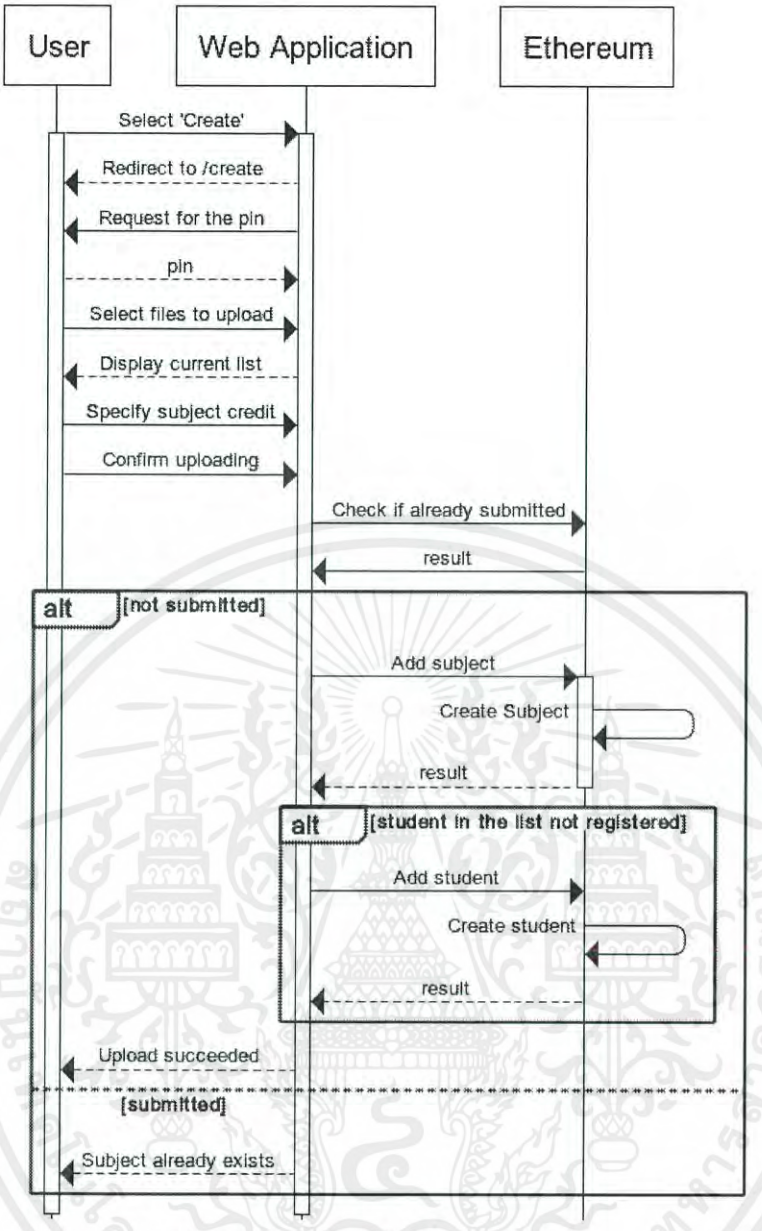
รูป 3.3 การตรวจสอบประเภทผู้ใช้งาน

การล็อคอินเข้าสู่ระบบเริ่มจากผู้ใช้งานเข้าระบบผ่านบราวเซอร์ บราวเซอร์จะติดต่อไปยัง Google เพื่อยืนยัน gmail ของผู้ใช้งาน เมื่อถูกต้อง จะติดต่อไปยังบล็อกเชนเซิฟเวอร์ว่ามีแอคเคาท์แอดเดรสของผู้ใช้งานนี้หรือยัง ถ้ายังจะทำการสร้างขึ้นมาใหม่และผู้ใช้งานต้องกำหนด PIN ขึ้นมาเพื่อใช้ปลดล็อคการทำงานส่วนอื่นๆ เมื่อตรวจสอบหรือสร้างแอคเคาท์แล้วผู้ใช้งานจึงจะเข้าสู่ระบบได้สำเร็จ



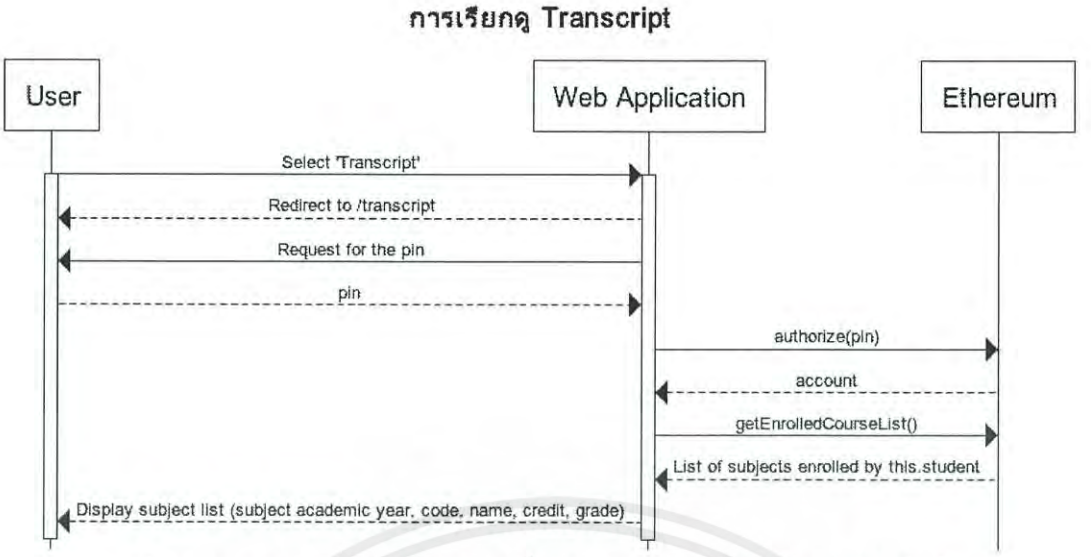
รูป 3.4 การเรียกดูวิชาที่อัปโหลดไปแล้วของอาจารย์

การเรียกดูวิชาที่อาจารย์ได้ทำการอัปโหลดขึ้นไปในบล็อกเชนจะเริ่มจาก อาจารย์กดลิงค์หรือเข้าหน้า Subject เว็บบราวเซอร์จะเปลี่ยนไปที่หน้าแสดงผลวิชาโดยจะต้องใส่ PIN เพื่อยืนยันตัวตน เมื่อ PIN ถูกต้องเว็บเซิฟเวอร์จะเรียกดูข้อมูลรายละเอียดวิชาของอาจารย์จากบล็อกเชน และแสดงผลทางหน้าเว็บ เมื่อกดเลือกเข้าไปดูรายละเอียดของวิชานั้น ๆ จะมีรายละเอียดของนักศึกษาที่ลงทะเบียนวิชานี้ไป เช่น ชื่อ นามสกุล รหัสนักศึกษา ผลการเรียน



รูป 3.5 การอัปโหลดไฟล์ของอาจารย์

การอัปโหลดไฟล์วิชาเข้าสู่ระบบโดยอาจารย์เริ่มจาก อาจารย์กดลิงค์หรือเข้าหน้า Course เว็บบราวเซอร์จะเปลี่ยนไปที่หน้าอัปโหลดคะแนน อาจารย์จะเลือกไฟล์ exel ที่เก็บรายละเอียดวิชาและคะแนนของนักศึกษาไว้ กดยืนยัน และเลือกหน่วยกิตของวิชา กดอัปโหลด โดยจะต้องใส่ PIN เพื่อยืนยันตัวตน เมื่อ PIN ถูกต้อง เว็บบราวเซอร์จะติดต่อไปยังบล็อกเชนเซิฟเวอร์และตรวจว่ามีวิชานี้ในระบบหรือยัง ถ้ามีแล้วจะอัปโหลดไม่สำเร็จ ถ้ายังไม่มีจะเข้าไปจัดการข้อมูลและส่งไปยังเซิฟเวอร์บล็อกเชนเก็บลงสู่ระบบ และขั้นตอนการเก็บจะตรวจสอบด้วยว่ามีนักศึกษาอยู่ในระบบหรือยัง ถ้ายังจะทำการเพิ่มข้อมูลของนักศึกษาลงไปด้วย เมื่ออัปโหลดสำเร็จจะขึ้นแจ้งเตือนที่หน้าเว็บ



รูป 3.6 การดูผลการเรียนในรายวิชาที่เรียนและอาจารย์อัปโหลดไฟล์ไปแล้ว

การเรียกดูผลคะแนนแต่ละวิชาของนักศึกษาเริ่มจาก นักศึกษากดลิงค์หรือเข้าหน้า Transcript เว็บบราวเซอร์จะเปลี่ยนไปที่หน้าแสดงผลคะแนนวิชาต่าง ๆ ของนักศึกษา โดยจะต้องใส่ PIN เพื่อยืนยันตัวตน เมื่อ PIN ถูกต้อง เว็บเซิฟเวอร์จะเรียกดูข้อมูลรายละเอียดคะแนนของนักศึกษาจากบล็อกเชน โดยจะมีชื่อวิชา ปีการศึกษา และผลการเรียน

3.4 ภาพรวมระบบ

3.3.1 User Machine

3.3.1.1 Web Browser

แสดงส่วนหน้าเว็บแอปพลิเคชัน สามารถรับส่งข้อมูลไปยัง Web Server

3.3.2 Web Server

3.3.2.1 Web Application

พัฒนาด้วย Vue.js ในการสร้างส่วนต่าง ๆ ของแอปพลิเคชัน และใช้ Web3.js เป็นส่วนที่ติดต่อรับและส่งข้อมูลไปยัง Blockchain Server โดยมี NGINX เป็น Server

3.3.2.2 Identity Provider

ใช้บริการของ Google โดยใช้ E-mail ของสถาบันฯเพื่อพิสูจน์ตัวตนว่าเป็นอาจารย์หรือนักศึกษาของสถาบันฯ โดยแต่ละจะมีสิทธิ์ในการเข้าถึงข้อมูลที่แตกต่างกัน

3.3.3 Blockchain Server

3.3.3.1 Ethereum Network

เครือข่ายของโหนด Ethereum เชื่อมต่อกันผ่าน Bootnode

1) Ethereum Virtual Machine พัฒนาด้วยภาษา Go เป็น Server ของ Ethereum Blockchain สามารถจำข้อมูลเข้าไปและเรียกออกมาแสดงผล และสามารถถูกเรียกใช้ผ่าน Web3.js

2) Decentralized Applications (DApp) หรือ Smart Contract พัฒนาด้วย Solidity เป็นส่วนที่ประมวลผล Business Logic ข้อมูลที่อยู่ในบล็อกเชน

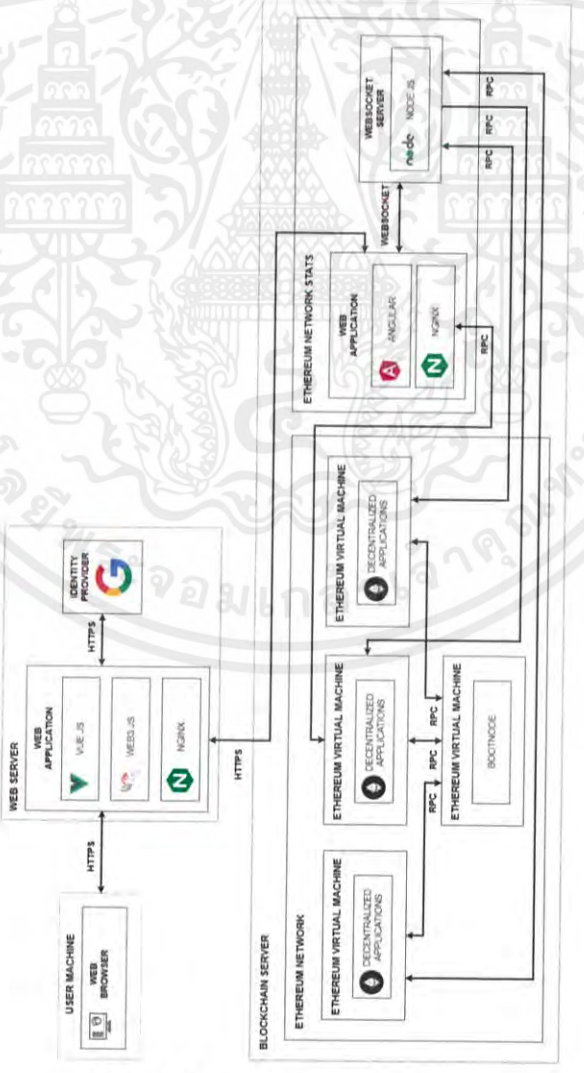
3) Bootnode เป็นตัวกลางเพื่อให้โหนด Ethereum Virtual Machine มองเห็นกันและติดต่อกันได้

3.3.3.2 Ethereum Network Stats

แสดงผลข้อมูลโหนดในระบบเครือข่าย

1) Web Application พัฒนาด้วย Angular แสดงผลผ่าน Browser

2) WebSocket Serverพัฒนาด้วย Node.js โดยติดต่อกับโหนด Ethereum Virtual Machine แล้วส่งข้อมูลไปยัง Web Application ผ่าน WebSocket



รูป 3.7 ผังการทำงานของระบบ

3.5 Data Dictionary

รูปแบบฐานข้อมูลไม่ใช่ทั้ง SQL และ noSQL โมเดลเป็นแบบ Append-Only ไม่สามารถ Modify Insert Delete ได้ ข้อมูลที่เข้ามาไม่มีรูปแบบ Transaction ไม่มีการเรียง แต่จะเรียงจากลำดับการถูก Validate แต่ละ Block โดย Smart Contract ใช้วิธี Hash Table ในการเก็บข้อมูลลงไป และยังไม่มีโมเดลรองรับ

ตาราง 3.5 Data Dictionary ของ Teacher

Field Name	Data Type	Field Size	Description	Example
addr	address	42	address ของอาจารย์	0x72ba7d8e73fe8eb666ea66babc8116a41bfb10e2
email	bytes32	32	อีเมล์ของอาจารย์	somchai.hh@gmail.com
teachers	bytes32	32	ชื่อผู้ใช้ของอาจารย์	somchai.hh

ตาราง 3.6 Data Dictionary ของ Course

Field Name	Data Type	Field Size	Description	Example
name	bytes32	32	ชื่อวิชา	Computer Programming
lecturer	bytes32	32	ชื่ออาจารย์ผู้สอน	somchai.hh
code	bytes8	8	รหัสวิชา	01000001
credit	uint	1	หน่วยกิต	3
academicYear	bytes8	8	ปีการศึกษา	2/2561
enrolledStdList	array	8 * Number of Students	รหัสนักเรียนที่ลงทะเบียน	58000001, 58000002
courses	course (struct)	81 + 8 * Number of Students	วิชา	Computer Programming, somchai.hh, 01000001, 3, 2/2561, { 58000001, 58000002 }
courseList	array	8 * Number of Courses	รายชื่อวิชาทั้งหมด	01000001, 01000002

ตาราง 3.7 ตาราง Dictionary ของ Student

Field Name	Data Type	Field Size	Description	Example
addr	address	42	address ของนักศึกษา	0x72ba7d8e73fe8eb666ea66babc8116a41bfb10e2
email	bytes32	32	อีเมล์ของนักศึกษา	sompong.rr@gmail.com
fname	bytes32	32	ชื่อของนักศึกษา	Sompong
lname	bytes32	32	นามสกุลของนักศึกษา	Rukrean
code	bytes8	8	รหัสนักศึกษา	58000001
credit	uint	1	หน่วยกิตสะสม	3
gpax	uint	5	ผลการเรียนเฉลี่ย	40000
enrolledCourseList	array	32 * Number of Courses	รายชื่อวิชาที่ลงทะเบียน	01000001, 01000002
grade	uint	5	ผลการเรียนสะสม	40000
students	student (struct)	152 + 32 * Number of Courses	นักเรียน	Sompong, Rukrean, 58000001, 3, 40000, { 01000001, 01000002 }, 40000
stdList	array	8 * Number of Students	รายชื่อนักเรียนทั้งหมด	58000001, 58000002

บทที่ 4
วิธีการดำเนินงานและผลการดำเนินงาน

4.1 ส่วนแสดงผลทางหน้าเว็บแอปพลิเคชัน และติดต่อกับบล็อกเชนผ่านทาง Web3.js

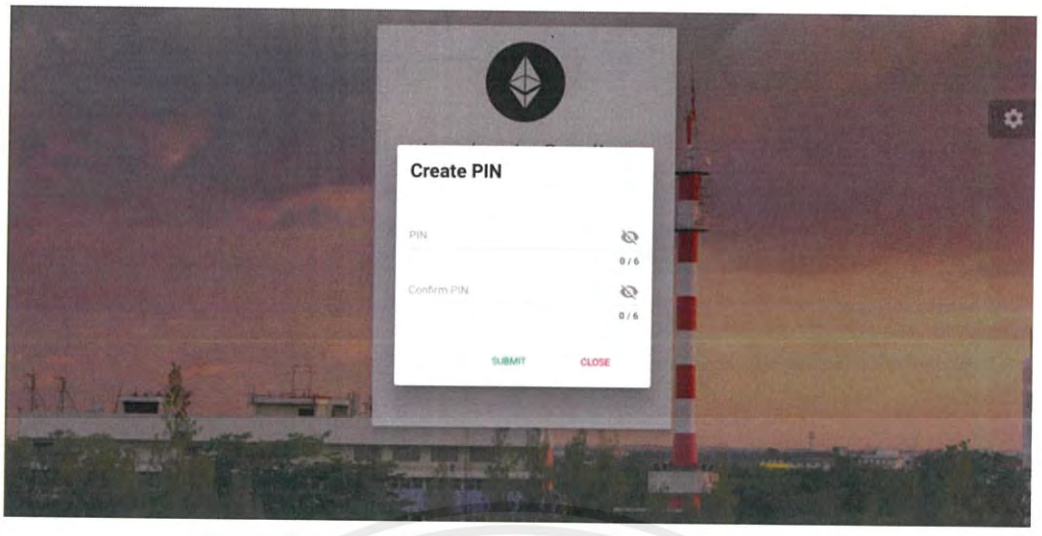
เป็นการพัฒนาในส่วนของการแสดงผลทางเว็บแอปพลิเคชัน โดยมีการส่งและรับข้อมูลจาก Blockchain ผ่านการเรียกใช้ Web3.js ซึ่งช่วยให้เข้าถึง และส่งข้อมูลได้สะดวกขึ้น ข้อมูลที่ส่งไปจะถูกแปลงเป็นฐาน 16 แล้วส่งไปยังบล็อกเชนและเมื่อจะนำข้อมูลที่อยู่ในบล็อกเชนมาใช้ก็ต้องแปลงจากฐาน 16 มาเป็นข้อความปกติ แล้วแสดงออกทางเว็บแอปพลิเคชัน

การใช้งานแบ่งเป็น 2 ฝ่าย คือ อาจารย์และนักศึกษา โดยแยกจาก E-mail ของผู้เข้าใช้งาน ซึ่งก็จะมีหน้าที่ที่แตกต่างกันไป

4.1.1 หน้าต่าง Login ของเว็บแอปพลิเคชัน

ในหน้านี้จะให้ผู้ใช้ลงชื่อเข้าใช้โดยใช้ E-mail ของสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ซึ่งระบบจะทำการตรวจสอบว่าผู้ใช้งานอยู่ในระบบ Registration System หรือไม่ ในกรณีที่ผู้ใช้ยังไม่ ในกรณียังไม่มีผู้ใช้งานในระบบ ระบบจะทำการเพิ่มผู้ใช้งานเข้าสู่ระบบ และจะให้ผู้ใช้กำหนด PIN เป็นตัวเลขความยาว 6 ตัว ระบบจะใช้ PIN ในการสร้าง Address ของผู้ใช้คนนั้น ๆ ซึ่งถ้าผู้ใช้เป็นอาจารย์จะทำการส่ง Gas ให้ เพื่อใช้ในการส่ง Transaction ในขั้นตอนการอัปโหลดไฟล์ แต่ถ้าผู้ใช้เป็นนักศึกษาก็จะไม่มีการส่ง Gas ให้เนื่องจากผู้ใช้ที่เป็นนักศึกษาไม่มีการส่ง Transaction ในกรณีที่ผู้ใช้เคยลงชื่อเข้าใช้แล้ว จะเข้าสู่หน้า Home ของเว็บแอปพลิเคชัน

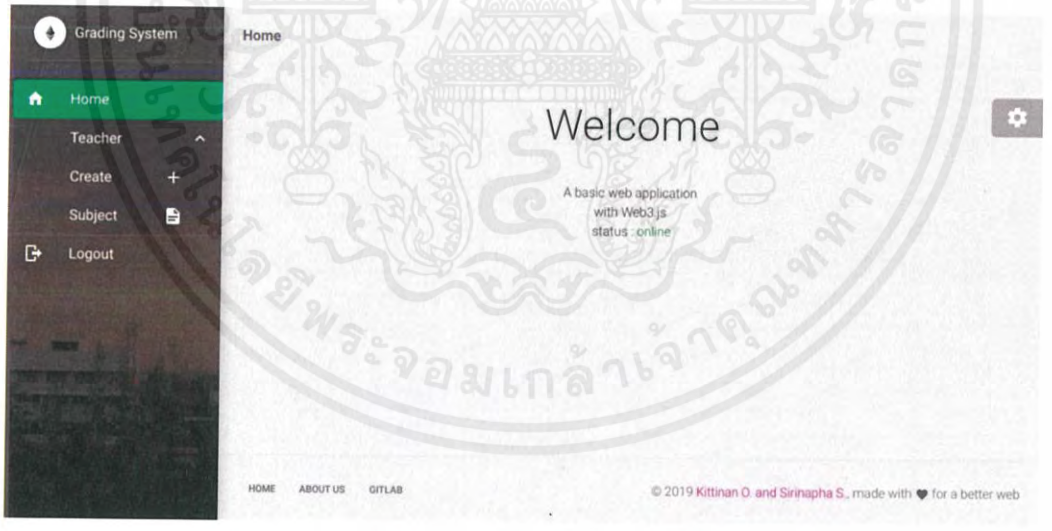
รูป 4.1 หน้าต่าง Login



รูป 4.2 สร้าง PIN สำหรับการใช้งานภายในเว็บแอปพลิเคชัน

4.1.2 หน้า Home ของอาจารย์

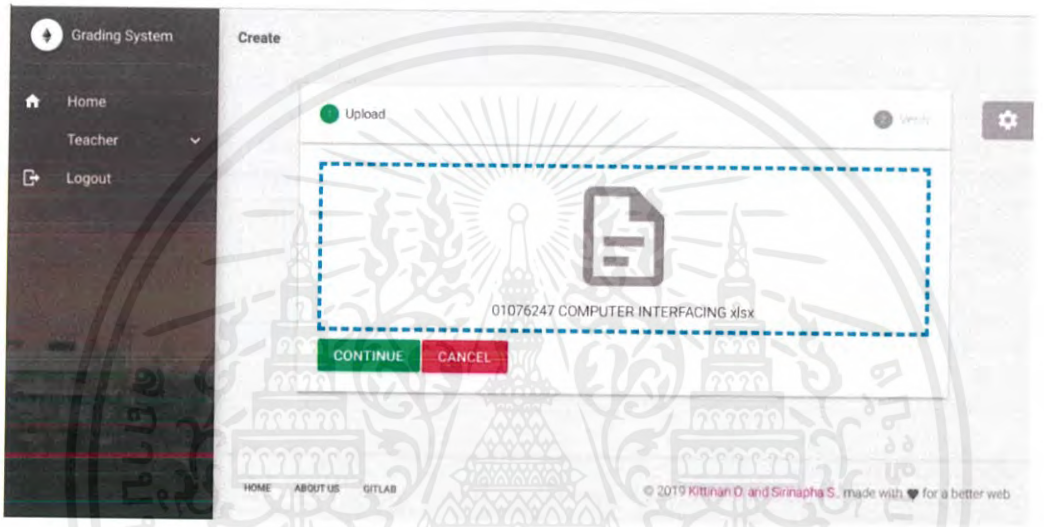
ในหน้านี้เป็นหน้าแรกที่อาจารย์ Login เข้ามาถึง หน้านี้จะแสดงสถานะของ Server ว่าพร้อมใช้งานหรือไม่ และแถบด้านข้างจะระบุรายการที่อาจารย์สามารถทำได้ ประกอบด้วย หน้าอัปโหลดไฟล์ (Create) และหน้าสำหรับดูข้อมูลวิชา (Subject)



รูป 4.3 หน้า Home ของอาจารย์

4.1.3 หน้า Create ของอาจารย์

เมื่อเลือกเมนู Create เข้ามาถึงจะมีกรอบที่สามารถลากไฟล์จากภายนอกเข้ามาใส่ได้ ซึ่งในหน้านี้อาจารย์จะทำการอัปโหลดไฟล์คะแนนของตนเองเข้าระบบ กำหนดค่าเครดิตของวิชาที่ต้องการอัปโหลด และต้องใส่ PIN ให้ตรงกับ PIN ที่สร้างไว้ในตอน Login ครั้งแรก โดยการอัปโหลดนั้นจะทำการส่ง Transaction โดยมีข้อมูลตามที่ได้อัปโหลดมา ซึ่งจะส่งจาก Address ของอาจารย์คนนั้น ๆ ไปยัง Blockchain Server การส่งแต่ละครั้งจะมีการใช้ Gas ที่อาจารย์แต่ละคนมีอยู่ในการส่ง ซึ่ง Gas เหล่านี้อาจารย์จะได้ตั้งแต่ครั้งแรกที่ทำการ Login



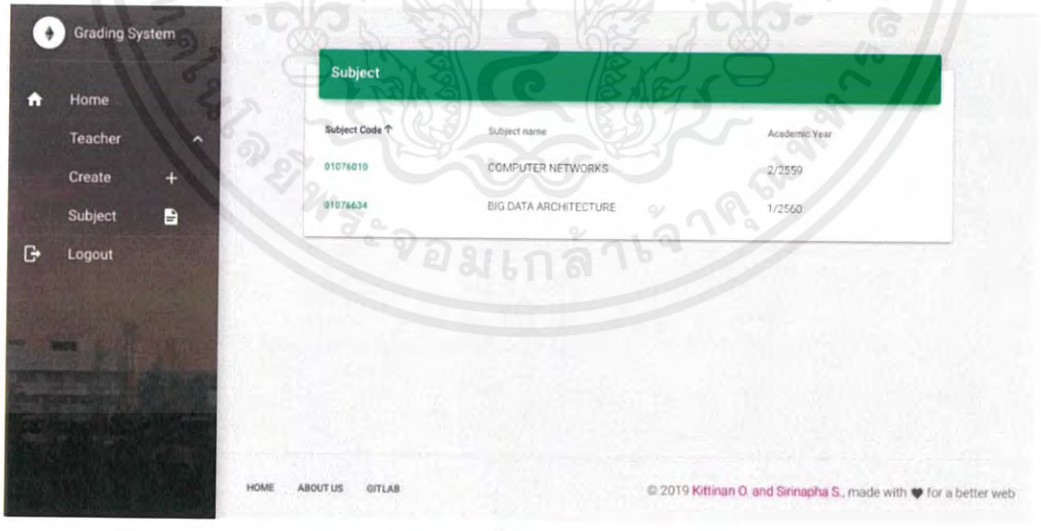
รูป 4.4 หน้า Create ของอาจารย์

รูป 4.5 หน้า Create ของอาจารย์ โดยสามารถเลือกเครดิตให้แต่ละวิชาได้

รูป 4.6 การใส่ PIN ก่อนการอัปโหลดไฟล์

4.1.4 หน้า Subject ของอาจารย์

เมื่อเลือกเมนู Subject ในหน้านี้อาจารย์จะสามารถเลือกดูรายละเอียดตามรายวิชาที่ตนเองอัปโหลดไฟล์ไปแล้ว ซึ่งจะต้องกรอก PIN ก่อนที่จะเข้ามาหน้านี้ เพื่อเป็นการยืนยันตัวตนอีกครั้ง ข้อมูลที่นำมาแสดงจะนำมาจากข้อมูลที่ถูกนำเข้าไปใน Blockchain Server แล้ว ในหน้าแรกจะขึ้นภาพรวมของวิชาทั้งหมดที่ถูกอัปโหลดไป เมื่อเลือกรายวิชาที่ต้องการดูรายละเอียดก็จะแสดงรายชื่อนักศึกษาในวิชานั้น และผลการเรียนที่ได้ของนักศึกษาคนนั้น



รูป 4.7 เลือกวิชาที่ต้องการดูรายละเอียด

รูป 4.8 รายละเอียดนักศึกษาและผลการเรียน

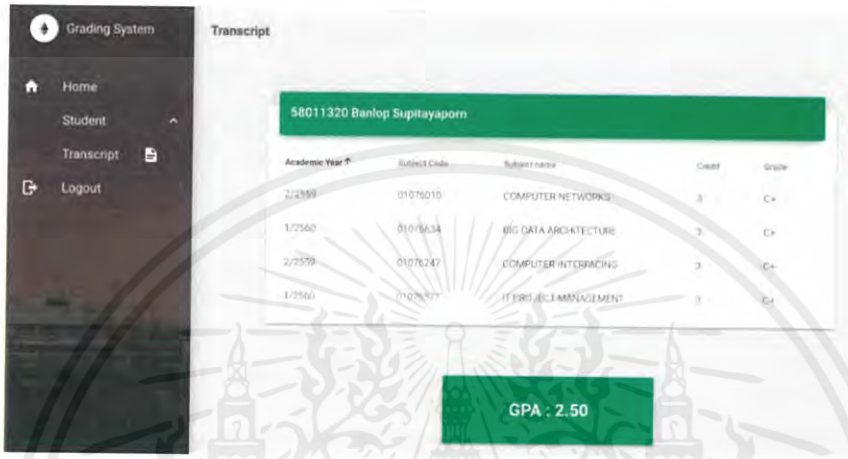
4.1.5 หน้า Home ของนักศึกษา

ในหน้านี้เป็นหน้าแรกที่นักศึกษา Login เข้ามาถึง ในหน้านี้จะแสดงสถานะของ Blockchain Server ว่าพร้อมใช้งานหรือไม่ และแถบด้านข้างจะระบุรายการที่นักศึกษาสามารถทำได้ ประกอบด้วย หน้าสำหรับดูผลการเรียนในรายวิชาที่เรียนและอาจารย์อัปโหลดไฟล์ไปแล้ว

รูป 4.9 หน้า Home ของนักศึกษา

4.1.6 หน้า Transcript ของนักศึกษา

ในหน้านี้นักศึกษาจะสามารถดูผลการเรียนของตัวเองที่อาจารย์อัปโหลดไปแล้ว ซึ่งจะต้องกรอก PIN ก่อนที่จะเข้ามาหน้านี้ เพื่อเป็นการยืนยันตัวตนอีกครั้ง ข้อมูลที่นำมาแสดงจะนำมาจากข้อมูลที่ถูกนำเข้าไปใน Blockchain Server แล้ว และจะแสดงผลการเรียนเฉลี่ยจากผลการเรียนทั้งหมดที่ได้รับมาด้วย



รูป 4.10 หน้า Transcript

4.2 การติดตั้งและตั้งค่าเว็บเซิฟเวอร์

ที่ฝั่งของ Front-end ใช้ NGINX เพื่อเป็นเว็บเซิฟเวอร์เพื่อโฮสไฟล์ของ Vue.js ที่ถูก Build ด้วย Webpack และทำการ Minify แล้วโดยที่ NGINX มีการติดตั้ง SSL เพื่อทำให้เชื่อมต่อผ่านโปรโตคอล HTTPS ด้วย Let's encrypt โดยใช้โดเมนชื่อ gradingsys.ml จาก freenom.com และตัวเว็บเซิฟเวอร์ถูกโฮสไว้ที่ Google Cloud และใช้ DNS ของ Google DNS

ส่วนในฝั่งของเซิฟเวอร์บล็อกเชน ใช้ NGINX เพื่อเป็นเว็บเซิฟเวอร์ของ Ethereum Net Stats และยังทำ reverse proxy ไปยัง Ethereum โหนดแต่ละโหนด และทำ Reverse Proxy ไปยัง WebSocket แต่ละตัวที่เชื่อมต่ออยู่แต่ละ Ethereum โหนด โดยใช้โดนเมนชื่อ regsys.ml และมีการติดตั้ง SSL ด้วย Let's encrypt เช่นเดียวกันกับเซิฟเวอร์ Front-end

4.3 ส่วนการประมวลผล และแสดงผลข้อมูลของโหนดใน Blockchain

4.2.1. Ethereum Network

เป็นการพัฒนาส่วนของการเก็บข้อมูลและประมวลผลผ่าน Smart Contract ซึ่งจะถูกเรียกใช้โดยเว็บแอปพลิเคชัน โดย Smart Contract จะถูกเก็บอยู่ใน Ethereum Blockchain

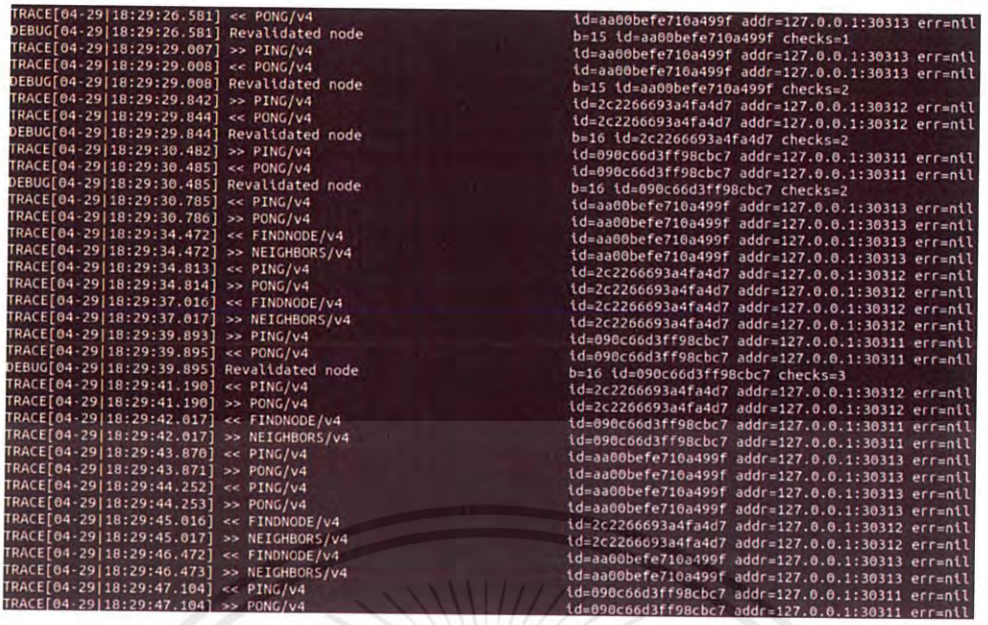
4.2.1.1 การ Deploy Smart Contract ลงไปใน Blockchain

ใช้ Truffle เป็นเครื่องมือในการ Deploy โดยนำ Solidity ของ Contract ทั้งหมดมาเขียน Script สำหรับ Deploy โดยใช้ JavaScript

รูป 4.11 Deploy Contract ลง Blockchain

4.2.1.2 การเชื่อมต่อโหนดต่าง ๆ เข้าด้วยกัน

เริ่มการทำงานของ Bootnode โดยกำหนด address และ Port เพื่อให้โหนดต่าง ๆ เชื่อมต่อเข้ามา หลังจากนั้น เมื่อรันโหนดต่าง ๆ ต้องกำหนด parameter เชื่อมต่อมายัง Bootnode ด้วย



รูป 4.12 Bootnode แสดงรายละเอียดของโหนดที่เชื่อมต่ออยู่

4.2.2. Ethereum Network Stats

เป็นการแสดงผลข้อมูลของโหนดด้วย Web Application บน Ethereum Network Stats Server ผ่านทาง WebSocket ที่จะส่งข้อมูลมาให้ตลอดเวลา

รูป 4.13 Web Application ที่แสดงผลข้อมูลของโหนดต่าง ๆ ในระบบ

4.4 ส่วนการเก็บข้อมูลในบล็อกเชน

บล็อกเชนจะเก็บข้อมูลในรูปแบบของ HEX ในฟิลด์ Data ของแต่ละ Transaction โดยการนำข้อมูลลงไปจะผ่าน Smart Contract เป็นตัวจัดการ ซึ่ง Smart Contract นั้นทำงานโดยการแปลงคำสั่งจากภาษา Solidity เป็น Assembly และจะถูกอ่านโดย Ethereum Console ซึ่งเป็น Runtime Environment

Solidity นั้นมีโครงสร้างคล้ายภาษา Java โดยจะมี Data field, Constructor, function (method), Inheritance และมีการกำหนด Visibility ของข้อมูลและฟังก์ชันได้ และยังมีจุดเด่นคือ Data type

ประเภท address และความสามารถในการ mapping ข้อมูลสองชนิด คล้ายกับการทำงานของ Key-Value ของ Dictionary ในภาษา Python และมีฟังก์ชันประเภท Assert, Revert, Require ที่ใช้เพื่อตรวจสอบว่าฟังก์ชันจะทำได้สำเร็จหรือไม่ เนื่องจากการทำงานนั้นจะต้องใช้ gas ตามกำหนด เมื่อ gas หมด ฟังก์ชันจะหยุดทำงาน หาก input ที่ป้อนเข้ามาไม่ผ่านเงื่อนไขของ ฟังก์ชันเหล่านี้จะถูกใช้ gas จนหมดและหยุดการทำงาน

โดยในระบบนี้จะแบ่ง Smart Contract ออกเป็นหลายโมดูลเพื่อใช้จัดการกับข้อมูลแต่ละประเภท คือ AccountManager, Teacher, Course, Student, Enrollment, Migration ซึ่งแต่ละส่วนจะมีการทำงานดังนี้

4.4.1 AccountManager

มีหน้าที่ในการจัดการ user ทั้งหมด มีการสมัครสมาชิก การตรวจสอบว่ามี user ในระบบหรือไม่ การ Map Gmail กับ Account Address ใน Ethereum

4.4.2 Teacher

มีหน้าที่ในการจัดการ User ที่เป็นอาจารย์ มีสิทธิ์ในการเพิ่มรายวิชาที่สอนและเรียกดูข้อมูลวิชาของตนเอง

4.4.3 Course

มีหน้าที่ในการจัดเก็บรายวิชาต่าง ๆ และรายละเอียดนักศึกษาในวิชานั้น ๆโดยจะมีโครงสร้างรายวิชาดังนี้ name, lecturer, code, credit, academicYear, enrolledStdList โดยการเพิ่มข้อมูลลงไปจะต้องเรียกฟังก์ชัน Authorize ด้วย Gmail ของอาจารย์เท่านั้น จึงจะสามารถใช้งานฟังก์ชันการเก็บข้อมูลได้ ตัวอย่าง การเรียกใช้ฟังก์ชันด้วย Input somkiat.tk@gmail.com

โปรแกรม 4.1 ฟังก์ชัน authorize

```
function authorize(bytes32 _email) public {
        require(_email != 0x0, "Not authorize!");
        addr = accMan.accounts(_email);
        require(addr != address(0x0), "No matched
account");
        email = _email;
    }
```

แล้วหลังจากนั้น เรียกใช้ฟังก์ชัน addCourse ด้วย input ของวิชานั้นลงไป

โปรแกรม 4.2 ฟังก์ชัน addCourse

```
function addCourse(
        bytes32 name,
        bytes32 lecturer,
        bytes8 code,
        uint credit,
        bytes8 academicYear) public {
            bytes32 tName = teacher.teachers(addr);
            require(tName == lecturer); // Check if
teacher is the owner of the course.
     course storage cors = courses[code];
            require(cors.code == 0x0); // Check if
course not already added.

            cors.name = name;
            cors.lecturer = lecturer;
            cors.code = code;
            cors.credit = credit;
            cors.academicYear = academicYear;
            courseList.push(code);
    }
```

โดยภายในฟังก์ชัน addCourse จะมีฟังก์ชัน require(tName == lecturer) แทรกอยู่ เพื่อเป็นการตรวจสอบว่า ชื่อของอาจารย์นั้นตรงกับอาจารย์ในวิชานั้น ถึงจะทำการเพิ่มคะแนนเข้าไปได้ โดยตัวแปร tName นั้นมาจาก bytes32 tName = teacher.teachers(addr); ซึ่ง parameter addr นั้นจะได้จากการ Authorize ด้วย Gmail ของอาจารย์คนนั้น ๆ จาก addr = accMan.accounts(_email);

4.4.4 Student

มีหน้าที่ในการจัดการ user ที่เป็นนักศึกษา และมีข้อมูลรายละเอียดของนักศึกษา รายวิชาที่ลงทะเบียน และการคำนวณผลการเรียน หากเป็นอาจารย์จะสามารถใส่ข้อมูลของนักศึกษาได้จากไฟล์คะแนน หากเป็นนักศึกษาสามารถเรียกข้อมูลและคะแนนได้ ตัวอย่าง การเพิ่มข้อมูลนักศึกษาลงไปในระบบ จะต้องทำการ Authorize ด้วย Gmail ของอาจารย์

โปรแกรม 4.3 ฟังก์ชัน authorize

```
function authorize(bytes32 _email) public {
        require(_email != 0x0, "Not authorize!");
        addr = accMan.accounts(_email);
        require(addr != address(0x0), "No matched
account");
        email = _email;
        bytes memory _code = new bytes(8);
        bytes8 out;
        for(uint i=0; i<8; i++){
            _code[i] = _email[i];
            out |= bytes8(_code[i] & 0xFF) >> (i * 8);
        }
        code = out;
    }
```

เรียกใช้ฟังก์ชัน addStudent ด้วย parameter ชื่อ นามสกุล รหัสนักศึกษา

โปรแกรม 4.4 ฟังก์ชัน addStudent

```
function addStudent(bytes32 fname, bytes32 lname,
bytes8 _code) public {
        bytes32 tName = teacher.teachers(addr);
        require(tName != 0x0, "No permission");
        student storage std = students[_code];
        require(std.fname == 0x0); // Check if student
not already added.
        std.fname = fname;
        std.lname = lname;
        std.code = _code;
        stdList.push(_code);
    }
```

ในฟังก์ชันจะมี require(tName != 0x0, "No permission"); เพื่อตรวจสอบว่าเป็นอาจารย์หรือไม่ โดย tName ได้จาก bytes32 tName = teacher.teachers(addr); และ addr มาจากการเรียกใช้ฟังก์ชัน authorize

4.4.5 Enrollment

มีหน้าที่ในการจัดการการลงทะเบียนวิชาเรียน Map วิชาและนักศึกษาเข้าด้วยกัน ตัวอย่าง

โปรแกรม 4.5 ฟังก์ชัน enroll

```
function enroll(bytes8 stdCode, bytes8 corsCode, uint
grade, uint credit) public {
        student storage std = students[stdCode];
        bytes32 tName = teacher.teachers(addr);
        bytes32 corsLecturer =
course.getLecturer(corsCode);
         require(tName == corsLecturer, "Teacher's name
not matched");
         for(uint i=0; i<std.enrolledCourseList.length;
i++) {
              if(std.enrolledCourseList[i] == corsCode)
{
                   revert("course already enrolled.");
              }
         }
         std.enrolledCourseList.push(corsCode);
         std.grade[corsCode] = grade;
            calcGpax(stdCode, credit, grade);
    }
```

ฟังก์ชัน enroll จะถูกเรียกใช้โดย Smart Contract สองตัวคือ Course และ Student โดยก่อนการ Enroll จะต้อง Authorize ด้วย Gmail ของอาจารย์ จาก require (tName == corsLecturer, "Teacher's name not matched"); และหากว่ามีวิชานี้อยู่ในระบบแล้วจะเพิ่มไม่สำเร็จ จาก revert("course already enrolled."); จะทำให้ Gas สำหรับ Smart Contract นี้ถูกใช้จะหมดและหยุดการทำงาน และเมื่อมีการเพิ่มวิชาเข้าไปรหัสวิชาจะถูกเก็บไว้ใน enrolledCourseList ของนักศึกษาที่ลงทะเบียนวิชานี้ จาก std.enrolledCourseList.push(corsCode); และใน Contract Coures ในทำนองเดียวกัน รหัสนักศึกษาจะถูกเก็บไว้ใน enrolledStdList จาก cors.enrolledStdList.push(stdCode); เช่นกัน

4.4.6 Migration

มีหน้าที่ในการเก็บประวัติการ Deploy Smart Contract ลงไปในบล็อกเชน

4.6 ส่วนการเรียกดูข้อมูลของนักศึกษา

เริ่มด้วยการเรียกใช้ฟังก์ชัน authorize ด้วย Gmail ของนักศึกษา

โปรแกรม 4.6 ฟังก์ชัน authorize

```
function authorize(bytes32 _email) public {
        require(_email != 0x0, "Not authorize!");
        addr = accMan.accounts(_email);
        require(addr != address(0x0), "No matched
account");
        email = _email;
        bytes memory _code = new bytes(8);
        bytes8 out;
        for(uint i=0; i<8; i++){
            _code[i] = _email[i];
            out |= bytes8(_code[i] & 0xFF) >> (i * 8);
        }
        code = out;
    }
```

โดยเมื่อ authorize แล้วจะได้ code ซึ่งคือรหัสนักศึกษา แล้วเรียกใช้ฟังก์ชั่น getStudent ด้วย parameter code คือ รหัสนักศึกษา

โปรแกรม 4.7 ฟังก์ชัน getStudent

```
function getStudent(bytes8 _code) public view returns (
          bytes32,
          bytes32,
          bytes8,
          uint,
          uint,
          bytes32[] memory) {
                student storage std = students[_code];
                bytes32 tName = teacher.teachers(addr);
                if(code == std.code || tName != 0x0) {
                    return (std.fname, std.lname,
std.code, std.credit, std.gpax,
std.enrolledCourseList);
                }
    }
```

ในฟังก์ชันจะมี if (code == std.code || tName != 0x0) เพื่อตรวจสอบว่า เป็นนักศึกษาคนนั้น ๆ หรือไม่ หรือเป็นอาจารย์หรือไม่ จึงจะมีสิทธิ์เรียกดูคะแนนได้

4.7 ส่วนการเรียกดูผลการเรียน

โปรแกรม 4.8 ฟังก์ชัน getGrade

```
function getGrade(bytes8 stdCode, bytes8 corsCode)
public view returns (uint) {
        student storage std = students[stdCode];
        bytes32 tName = teacher.teachers(addr);
        bytes32 corsLecturer =
course.getLecturer(corsCode);
        require(code == std.code || tName ==
corsLecturer, "No permission");
        return students[stdCode].grade[corsCode];
    }
```

การเรียกดูผลการเรียนจะมี parameter สองตัวคือ รหัสนักศึกษา โดยจะมี require(code == std.code || tName == corsLecturer, "No permission"); เพื่อตรวจสอบว่าเป็นนักศึกษาคนนั้นๆ หรือเป็นอาจารย์ประจำวิชาหรือไม่

โปรแกรม 4.9 ฟังก์ชัน calcGpax

```
function calcGpax(bytes8 _code, uint credit, uint
grade) private {
        student storage std = students[_code];
        uint gpax = std.gpax;
        uint totalCredit = std.credit;
        uint dividend =  gpax*totalCredit;
        uint divisor = totalCredit = std.credit +=
credit;
        dividend = dividend+(grade*credit);
        uint newGpax = dividend/divisor;
        std.gpax = newGpax;
    }
```

มี parameter รหัสนักศึกษา หน่วยกิตรวม ผลการเรียนที่ได้ โดยในภาษา Solidity จะไม่สามารถเก็บข้อมูลประเภท Floating-point ได้ จึงประยุกต์โดยการเพิ่มหลักของผลการเรียนแทน เช่น ผลการเรียน A = 4 จะแทนด้วย 40000 ซึ่งเสมือนการเก็บทศนิยม 4 หลัก เมื่อถูกเรียกใช้ที่เว็บเซิฟเวอร์จะหารด้วย 10000 เพื่อให้แสดงผลตามปกติ

โดยการคำนวณผลการเรียนรวมจะเรียกดูผลการเรียนสะสม หน่วยกิตสะสม และคำนวณด้วยผลรวมของผลคูณของหน่วยกิตรวมและผลการเรียนรวมและผลคูณของผลการเรียนและหน่วยกิตหารด้วย หน่วยกิตทั้งหมด จะได้ผลการเรียนรวมใหม่ทุกครั้งที่มีการเพิ่มวิชาเข้าไปในระบบแบบอัตโนมัติ

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุป

ภายในปริญญานิพนธ์มีการทำงานร่วมกัน 2 ส่วน คือ ส่วน Blockchain Server และส่วนของ Web Server ที่เป็นเว็บแอปพลิเคชัน

5.1.1 ส่วนของ Blockchain Server

สามารถ Deploy Smart Contract ลงไปในบล็อกเชนมีการส่งข้อมูลผ่าน WebSocket ไปยัง WebSocket Server ที่เป็นส่วนแสดงผลของโหนดต่าง ๆ ผ่านหน้าเว็บแอปพลิเคชัน และมีการรับและส่งข้อมูลผ่าน HTTPS ไปยัง Web Server ที่เป็นส่วนหน้าเว็บแอปพลิเคชันที่ติดต่อกับผู้ใช้งาน

5.1.2 ส่วนของ Web Server

สามารถแสดงผลของข้อมูลและส่งข้อมูลไปยังบล็อกเชนได้ โดยมีการใช้ Web3.js ในการรับและส่งข้อมูลเหล่านั้น ซึ่งการส่งข้อมูลเป็นการส่งแบบที่มี Async/Await เข้ามาช่วย เป็นการรอให้ทำการส่งข้อมูลชุดก่อนหน้าให้ Blockchain Server ให้เสร็จก่อน แล้วจึงค่อยส่งชุดถัดไป ส่วนของการเรียกดูข้อมูลนั้นสามารถทำงานได้เลย ไม่จำเป็นต้องใช้การส่งที่มี Async/Await เข้ามาช่วย แต่การเรียกดูข้อมูลจำนวนมากก็อาจต้องใช้เวลารอข้อมูลพร้อมช่วงหนึ่ง

5.2 ปัญหาและอุปสรรค

5.2.1 ส่วนของ Blockchain Server

ในส่วนของ Smart Contract ที่ใช้ Solidity ในการทำงานประมวลผลคำสั่งต่าง ๆ ยังเป็นเวอร์ชัน Beta อยู่ ดังนั้นในการทำงานบางครั้งจะมีปัญหาที่ไม่ทราบสาเหตุที่แน่ชัด เพราะยังไม่มีการทำ Error Handling มารองรับ

ในส่วนของการเก็บข้อมูลในรูปแบบ String ไว้ภายในโครงสร้างข้อมูลแบบ Struct ยังไม่รองรับการ Return ค่าจากฟังก์ชันใน Solidity ดังนั้นจึงต้องประยุกต์การเขียนโค้ดแยกย่อยการส่งข้อมูลออกมาทีละส่วนและใช้ข้อมูลประเภท bytes32 แทน String จึงทำให้เมื่อต้องการนำเข้าข้อมูลภาษาไทย ในกรณีที่มีความยาวเกินกว่า 32 bytes จะต้องแยกส่วนกันเพื่อเก็บข้อมูล ทำให้เกิดความยุ่งยากในการออกแบบโครงสร้างของ Smart Contract

5.2.3 ส่วนของ Web Server

Web3.js ในแต่ละเวอร์ชันจะมีรูปแบบการทำงานที่ไม่เหมือนกันทำให้บางครั้งเกิดความผิดพลาดที่มองไม่เห็น เช่น ความผิดพลาดจากฟังก์ชันที่มีการเปลี่ยนแปลง ทำให้ไม่สามารถใช้ชุดคำสั่งเหมือนเดิมแล้วจะได้ผลลัพธ์แบบเดิม

การส่งข้อมูลเป็นจำนวนมากจากหน้าเว็บไปยัง Blockchain Server นั้นไม่สามารถใช้การส่งแบบปกติได้ ต้องใช้การส่งที่มี Async/Await เข้ามาช่วยทำให้การอัปโหลดข้อมูลเพิ่มความล่าช้าขึ้น

5.3 แนวทางการดำเนินงานต่อ

1) ทำให้เว็บแอปพลิเคชั่นรองรับการรับส่งข้อมูลภาษาไทยได้อย่างสมบูรณ์

2) พัฒนาแอปพลิเคชันและฐานข้อมูลสำหรับเก็บ Wallet Object ในกรณีที่ผู้ใช้งาน เข้าใช้งานเว็บแอปพลิเคชันจากเครื่องคอมพิวเตอร์ที่ต่างกัน เพื่อความสะดวกของผู้ใช้งานไม่ต้องจดจำ Private key ของ Account ตัวเอง

3) การแก้ไขข้อมูลที่อาจารย์อัพโหลดไฟล์เอกสารได้จากหน้าเว็บแอพลิเคชั่นได้เลย และการบันทึกเอกสารและนำออกมาสู่เครื่องผู้ใช้ได้

4) การเพิ่มหน้าแสดงผลประวัติการอัพโหลดไฟล์เอกสาร และการอัพโหลดไฟล์เอกสารซ้ำได้

บรรณานุกรม

Annie Azaña. 2561. **A complete guide to building Ethereum DApps with MetaMask.** [Online].Available : https://medium.com/crowdbotics/building-ethereum-dapps-with-meta-mask-9bd0685dfd57

Pasupol Bunsaen. 2561. **Web3.js 101**. [Online].Available : https://medium.com/@njth/web3-js-101-563001bf24d1

Patrick Hubbard. 2561. **Smart Contracts are Awesome!**. [Online]. Available : https://blockgeeks.com/guides/smart-contracts/

Bill Hess. 2561. **What Is The Blockchain?.** [Online] .Available : https://pixelprivacy.com/resources/what-is-the-blockchain/

Simply Explained – Savjee. 2560. **How does a blockchain work - Simply Explained**. [Online] .Available : https://www.youtube.com/watch?v=SSo_EIwHSd4

Mahesh Murthy. 2560. **Full Stack Hello World Voting Ethereum Dapp Tutorial—Part 1**. [Online].Available : https://medium.com/@mvmurthy/full-stack-hello-world-voting-ethereum-dapp-tutorial-part-1-40d2d0d807c2

Lisk. 2561. **Consensus Protocols.** [Online].Available : https://lisk.io/academy/blockchain-basics/how-does-blockchain-work/consensus-protocols

Pramod Chandrayan. 2561. **Role Of A Consensus Protocol To Build Secure BlockChain Platform.** [Online].Available : https://medium.com/swlh/role-of-a-consensus-to-build-a-blockchain-system-537e7bef4709

Jiraboon Narktong. 2560. **Proof of Work vs Proof of Stake สิ่งที่นักขุดควรจะรู้ก่อนเริ่มขุด.** [Online].Available : https://siamblockchain.com/2017/08/13/proof-of-work-vs-proof-of-stake/

Anonymous. 2561. **Blockchain EP.4 : Proof of Work.** [Online]. Available : https://blog.nextzy.me/blockchain-ep-4-proof-of-work-575f306490f4