การทดสอบความเป็นไปได้การประยุกต์ใช้บล็อกเชนกับโฉนดที่ดินในกระบวนการจำนอง

Proof-of-Concept (PoC) of Land Mortgaging Process in Blockchain-based Land Registration System of Thailand

นายชนพ คอนนาสี

Chanop Khonnasee

นายสวิตต์ เลิศพัฒนศักดิ์

Swiss Lertpattanasak

โครงงานนี้เปนสวนหนึ่งของการศึกษาหลักสูตรวิทยาศาสตรบัณฑิต

ภาควิชาวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ ปีการศึกษา พ.ศ. 2561

คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

โครงงาน	การทดสอบความเป็นไปได้การประยุกต์ใช้บล็อกเชนกับโฉนดที่ดินในกระบวนการจำนอง Proof-of-Concept (PoC) of Land Mortgaging Process in Blockchain-based Land Registration System of Thailand		
นิสิต	นายชนพ	คอนนาสี	58102010803
	นายสวิตต์	เลิศพัฒนศักดิ์	58102010828
ปริญญา	วิทยาศาสตร์บัณฑิต(วท.บ.)		
ภาควิชา	วิทยาการคอมพิวเตอร์		
อาจารย์ที่ปรึกษาโครงงาน	ผู้ช่วยศาสตราจารย์ ดร.จันตรี ผลประเสริฐ		

ลงชื่อ.....

(ผู้ช่วยศาสตราจารย์ ดร.จันตรี ผลประเสริฐ)

อาจารย์ที่ปรึกษาโครงงาน

บทคัดย่อ

โฉนดที่ดินเป็นเอกสารสำคัญที่สามารถใช้ในการตรวจสอบความเป็นเจ้าของและประวัติการทำธุรกรรมของที่ดิน ในประเทศไทยกรมที่ดินทำหน้าที่ดำเนินการและอำนวยความสะดวกในการทำธุรกรรมต่างๆ ที่เกี่ยวข้องกับโฉนดที่ดิน เทคโนโลยีในปัจจุบันที่ใช้ในการดำเนินงานมักประสบปัญหา เรื่องความน่าเชื่อถือของการดำเนินงานและความซับซ้อนที่ทำให้เกิดความล่าช้า บทความนี้ได้ปรับปรุงกระบวนการจำนองโฉนดที่ดินของกรมที่ดินโดยใช้ blockchain เพื่อปรับปรุงความน่าเชื่อถือของการจัดเก็บและลดขั้นตอนในกระบวนการการทำสัญญาจำนองกับธนาคาร โดยใช้ Private blockchain ของ Ethereum blockchain ในการพัฒนาและทดสอบผลการทดลองแสดงให้เห็นว่าระบบที่เราเสนอสามารถจัดการธุรกรรมรายการต่อวินาทีในช่วงจำนวนธุรกรรมที่น่าสนใจผลลัพธ์แสดงแนวโน้มที่มีศักยภาพทั้งใน การลดกระบวนการและประสิทธิภาพของ blockchain สำหรับกรณีการใช้งานที่เสนอ

Abstract

Land title deeds are important documents that can be used to verify ownership and transfer history of lands. In Thailand, the Department of Lands operates and facilitates the transfer of ownership and other operations related to lands. The current technology used in the operations often encounters problems with the reliability of operations and complexities that cause delays. This paper investigates the proof-of-concept (PoC) of the integration of blockchain in the land registration system to improve the reliability of storage and reduces the steps in the process. The process of loan contracts with banks is used in the PoC and an Ethereum-based private blockchain is developed and tested. Results show that our proposed system can handle transactions per second over a range of a number of transactions of interest. The results show promising potential both in the reduction of process and the performance of blockchain for the proposed use case.

กิติกรรมประกาศ

สารบัญ

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

จากกระบวนการจำนองที่ดินในปัจจุบัน การจำนองที่ดินกับธนาคารนั้นต้องผ่านการตรวจสอบสารบัญจดทะเบียนของโฉนดที่ดิน เพื่อตรวจสอบภาระก่อนการอนุมัติเงินจำนอง เมื่อธนาคารอนุมัติแล้วจึงตกลงวันเซ็นสัญญาที่กรมที่ดินที่อีกครั้ง จากตัวอย่างข้างต้นมักพบปัญหาคือ ความน่าเชื่อถือของโฉนดที่ดิน, กระบวนการดำเนินงานที่ต้องผ่านการตรวจสอบจากกรมที่ดิน ทำให้การทำธุรกรรมเกิดความล่าช้า และเกิดข้อผิดจากกระบวนการเก็บบันทึกข้อมูล

จากปัญหาข้างต้นทางผู้จัดทำเล็งเห็นถึงประโยชน์ของ Blockchain ที่เป็นเทคโนโลยีที่มีความถูกต้องของข้อมูลสูง เนื่องจากข้อมูลภายใน Chain ไม่สามารถแก้ไขได้ ดังนั้น Blockchain จะเข้ามาแก้ไขปัญหาความน่าเชื่อถือของโฉนดที่ดินไม่ให้เกิดการปลอมแปลงได้ และ Smart Contracts เข้ามาช่วยลดกระบวนการต่างๆที่มีบุคคลที่ 3 เข้ามาเกี่ยวข้องทำให้เกิดความยุ่งยาก ให้มีความสะดวกมากยิ่งขึ้น

รูปที่ 1 ตัวอย่างสารบัญจดทะเบียนของที่ดิน

http://www.softbizplus.com/knowledge-management/459-how-to-write-the-land-authorized

1.2 วัตถุประสงค์ของงานวิจัย

1. ศึกษาการใช้ Blockchain ในการเก็บบันทึกข้อมูลโฉนดที่ดิน
2. ศึกษาการใช้ Smart contract ในการลดกระบวนการจำนองของโฉนดที่ดิน
3. ทดสอบประสิทธิภาพของระบบที่นำเสนอภายใต้สภาพแวดล้อมที่หลากหลาย

1.3 ขอบเขตของโครงงาน

1. พัฒนา Private Blockchain เพื่อใช้ในการสร้างโครงสร้างการจัดเก็บข้อมูลที่เหมาะสมกับโฉนดที่ดิน
2. พัฒนา Smart Contracts เพื่อใช้ควบคุมจัดการโฉนดที่ดินและธุรกรรมต่างๆ
3. พัฒนา Web Application สำหรับกรมที่ดิน ประชาชน และธนาคาร เพื่อให้เข้าถึง Blockchain ได้
4. ทดสอบประสิทธิภาพของระบบเทียบกับข้อมูลทางสถิติของกรมที่ดิน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. เป็นการศึกษาเรียนรู้ Blockchain เพื่อประยุกต์ใช้ในกระบวนการปัจจุบันของกรมที่ดิน
2. เจ้าของโฉนดที่ดินสามารถยืนยันความเป็นเจ้าของโฉนดที่ดินได้
3. ผู้มีส่วนได้ส่วนเสียในที่ดินสามารถตรวจสอบความถูกต้องของโฉนดที่ดินได้
4. ลดกระบวนการดำเนินงานที่เกิดขึ้นกับโฉนด เมื่อเกิดธุรกรรม
5. สร้างความน่าเชื่อถือให้กับโฉนดที่ดินและธุรกรรมต่างๆ

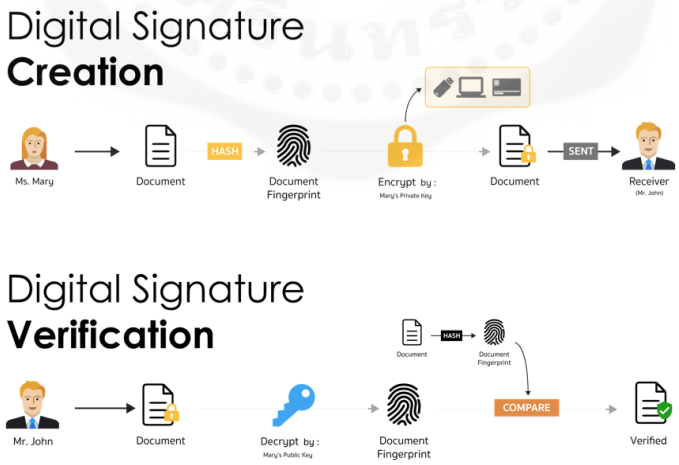
บทที่ 2
ทฤษฎีและแนวคิดที่ใช้ในการทำวิจัย

มีการประยุกต์ใช้ทฤษฎีต่างๆที่ช่วยในการสร้างความปลอดภัยและความน่าเชื่อถือให้โฉนดที่ดินและกระบวนการสัญญาจำนอง โดยจะมีการใช้ Digital signature ในการยืนยันตัวตนของผู้ใช้ในระบบ ที่ใช้งานผ่านทาง User Interface ใช้ Blockchain ในการสร้างเครือข่ายข้อมูลแบบกระจายอำนาจ และใช้ Smart Contract ในการจัดการข้อมูลของโฉนดที่ดินรวมถึงการกระบวนทำสัญญาจำนอง

2.1 ทฤษฎีเกี่ยวกับ Digital signature[1]

Digital Signature เป็นลายเซ็นที่อยู่ในรูปแบบของอิเล็กทรอนิกส์ ที่มีคุณสมบัติด้านความปลอดภัยเพื่อให้มีความน่าเชื่อถือมากยิ่งขึ้น ประกอบด้วย

- **Signer Authentication** เป็นความสามารถในการพิสูจน์ว่าใครเป็นคนเซ็นเอกสาร ตัวลายเซ็นจะสามารถใช้ในการเชื่อมโยงไปยังบุคคลที่เซ็นเอกสารได้
- **Data Integrity** เป็นความสามารถในการตรวจสอบ หรือพิสูจน์ได้ว่ามีการแก้ไขเอกสารหลังจากที่ได้มีการเซ็นไปแล้วหรือไม่
- **Non-repudiation** การไม่สามารถปฏิเสธความรับผิดชอบได้ เนื่องจากลายเซ็นที่สร้างขึ้นมีเอกลักษณ์ สามารถพิสูจน์ในชั้นศาลได้ว่าใครเป็นผู้เซ็นเอกสาร



รูปที่ 2 กระบวนการของ Digital Signature

https://www.bcircle.co.th/2017/09/30/digital-signature/

จากรูปที่ 2 แสดงถึงการสร้าง Digital Signature จากการนำเอกสารสิทธิของ User(Ms.Mary) มา Hash ให้ได้ DocumentFingerprint ซึ่งส่วนนี้ user จะเก็บไว้ แล้วจึงนำไป Ecryption ด้วย Public Key ของ User(Ms.Mary) จะทำให้ได้ Document ที่ผ่านการ Ecryption เก็บไว้ที่อีกฝ่ายที่ต้องการตรวจสอบข้อมูล (Mr.john) จากนั้นสามารถยืนยัน Digital Signature โดยการให้ User(Ms.Mary) ใช้ Private Key ในการ Decryption เอกสารที่อยู่ที่ Mr.john เพื่อยืนยัน DocumentFingerprint ว่าตรงกันเป็นการเสร็จสิ้นกระบวนการ

2.2 ทฤษฎีเกี่ยวกับ Blockchain

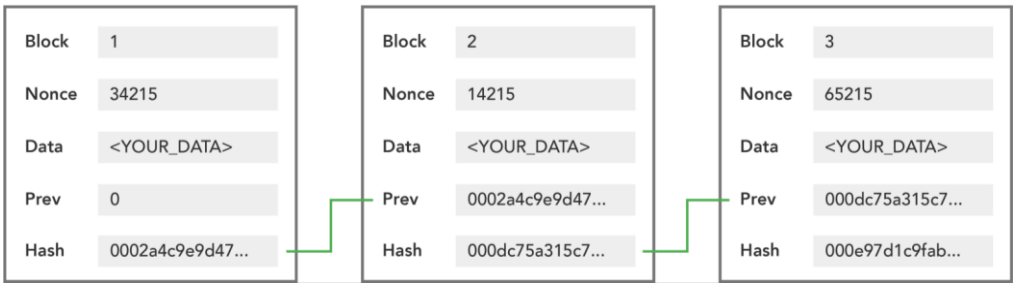
บล็อกเชนเป็นเทคโนโลยีในการจัดเก็บข้อมูลแบบไม่ต้องอาศัยคนกลาง (Third Party) ในการเก็บข้อมูล โดยทั่วไปการเก็บข้อมูลดิจิทัลนั้นจะต้องมีแม่ข่ายหรือศูนย์ข้อมูลกลาง (Data Center) ใน การจัดเก็บข้อมูล หากสังเกตดูจะพบว่าหลายๆ กิจการรอบตัวเราล้วนแล้วแต่ต้องอาศัยคนกลางทั้งสิ้น ไม่ว่าจะเป็น ธุรกรรมทางการเงินที่ต้องอาศัยธนาคารในการเป็นตัวกลางให้ธุรกรรมทางการเงินนั้นเสร็จสมบูรณ์ หรือการซื้อขาย ที่ดินก็ต้องพึ่งพากรมที่ดินในการเก็บข้อมูลและยืนยันความถูกต้องของโฉนด เมื่อเป็นเช่นนี้แล้วหากคนกลางหายไป รูปแบบกิจกรรมก็ย่อมเปลี่ยนไปด้วยเช่นกัน[12]

Blockchain จะเปลี่ยนรูปแบบการทำงาน ไปโดยสิ้นเชิง สมาชิกทุกคนมีสิทธิเท่ากันในการรับส่งข้อมูล ระบบนี้มีกลไกในการเข้ารหัสของข้อมูล ซึ่งการเข้ารหัสของข้อมูลนี้ก็เพื่อรักษาความปลอดภัยของข้อมูล หากข้อมูลมีการเปลี่ยนแปลงสมาชิกทุกคนในเครือข่ายต้องทำการตรวจสอบข้อมูลนั้นเพื่อยืนยันความถูกต้องของข้อมูล ดังนั้น ทุกๆการเปลี่ยนแปลง สมาชิกทุกคนจะได้รับทราบ ยืนยัน และจัดเก็บข้อมูลต่อไป

2.2.1 องค์ประกอบของ blockchain [2]

1. Block

เป็นการเก็บ Data Transaction โดยบรรจุในกล่อง (Block) และเมื่อปิดกล่องแล้ว โดยจะ Hash ข้อมูล Data Transaction ภายในกล่องกำกับไว้ที่ Block Header Block ทำให้ Data ไม่สามารถเปลี่ยนแปลงได้ และกระจายข้อมูลไปให้ทุกคนที่เกี่ยวข้องเก็บไว้



รูปที่ 3 องค์ประกอบภายใน Block

https://medium.com/dcen/block-data-in-blockchain-cddbb134eb36

จากรูปที่ 3 จะเห็นได้ว่าภายใน block จะมีองค์ประกอบดังนี้

- **Block number** คือ ตัวเลขระบุถึงลำดับก่อนหลังของกล่อง
- **Nonce** (Number Used One) คือ ค่าที่ใช้ในการค้นหาค่า Hash ของกล่องตามกฎที่ตั้งไว้ (Proof-of-Work) เช่น เมื่อต้องการสร้าง กล่อง เราต้องหาค่า Nonce ที่ทำให้ Hash ของ บล็อก มีค่าตามที่กำหนด ตัวอย่าง หาค่า Hash ขึ้นตนด้วย 0 จำนวน 4 หลัก และด้วยวิธีการนี้เอง ทำให การแก้ไขข้อมูลที่บันทึกลง กล่องที่อยู่ใน Chain แก้ไขได้ยาก
- **Data** คือ ข้อมูลที่อยู่ภายในกล่อง เช่น Transaction ต่างๆ
- **Hash** คือ วิธีการอย่างหนึ่งซึ่งทำให้ข้อมูลส่วนหนึ่งหรือทั้งหมด ให้กลายเป็นจำนวนเล็กๆ อันหนึ่ง อย่างมีปฏิสัมพันธ์ ซึ่งจำนวนดังกล่าวเปรียบได้ว่าเป็น "ลายนิ้วมือ" ของข้อมูล ซึ่งหากข้อมูลในการ Hash เปลี่ยนไปเพียงตัวเดียว ค่า Hash จะเปลี่ยนแปลงไปอย่างสิ้นเชิง
- **SHA 256** คือ เป็น algorithm ในการทำ Hash ซึ่งได้ผลลัพธ์ออกมา 256 bit
- **Previous Hash** คือ ค่า Hash ของกล่องก่อนหน้า

2. **Chain**

เป็นการนำกล่องที่สร้างขึ้นมาต่อกับกล่องก่อนหน้าด้วยการเก็บค่า Hash ไว้ในกล่องถัดไป หากต้องการแก้ Data Transaction ภายในกล่อง จะทำให้ค่า Hash ที่กำกับไว้เปลี่ยนไป ทำให้ต้องตามแก้ทั้ง Chain ดังนั้น Hash เปรียบเสมือนค่าที่นำมาใช้ในการยืนยันข้อมูลแต่ละกล่องว่ามีความถูกต้อง

3. Consensus

เป็นข้อตกลงรวมกันของผู้ที่อยู่ในเครือข่าย Blockchain นั้น ทำข้อตกลงในการใช้งานร่วมกัน เพื่อใช้ในการตรวสอบความถูกต้อง และปิดกล่องของข้อมูล โดยข้อตกลงนั้นมีหลายวิธีการ เช่น

- **Proof-of-work** : เป็นข้อตกลงในการพิสูจน์วาการทำธุรกรรมถูกต้อง โดยคนที่จะได้ปิดกล่องคือคนที่โชคดีหรือคนที่ขยันแก้โจทย์มากที่สุด

- **Proof-of-stake** : เป็นข้อตกลงในการพิสูจน์วาการทำธุรกรรมถูกต้อง โดยคนที่จะได้ปิดกล่องคือคนที่ถือครอง Value มากยิ่งมีสิทธิมากและอาจจะใส่กฎบางข้อเข้าไปเพื่อความยุติธรรมแก่คนอื่น โดยผู้ที่ทำหน้าที่ตรวจสอบเรียกว่า “Miner” หรือ “Validator”

2.2.2 ประเภทของ Blockchain [3]

ปัจจุบันแบ่ง บล็อกเชนเป็น 3 ประเภทคือ

1. public blockchain เป็นบล็อกเชนแบบสาธารณะที่ทุกคนสามารถเข้าใช้งานได้อย่างอิสระ เช่น Bitcoin, Ethereum เป็นต้น

ข้อดี การส่งข้อมูลไปให้หน่วยงานผู้รับปลายทางเราก็ไม่ต้องมาสร้างช่องทางส่งข้อมูลกัน หรือในตอนนี้เรานิยมทำ Web Service API คุยกัน องค์กรผู้ส่งข้อมูลเพียงแค่ใส่ข้อมูลลงไปใน Blockchainและจ่าหน้าซองข้อมูลถึงองค์กรผู้รับเท่านั้นผู้รับก็ได้รับข้อมูลไปโดยทันที

ข้อเสีย ข้อมูลที่ใส่เข้าไปบนโลก Public Blockchain นั้นจะกลายเป็นว่าข้อมูลนั้นโดนเปิดเผยแก่สาธารณะชน

2. private blockchain เป็นบล็อกเชนที่สร้างขึ้นเพื่อใช้ในองค์กร หรือบริษัทภายในเครือเท่านั้นที่มีสิทธิเข้าถึง เช่น Ripple (ให้บริการด้านการโอนเงินระหว่างประเทศ)

ข้อดี **ลดปัญหาความเป็นส่วนตัวของข้อมูล และ** สามารถปรับกฎเกณฑ์ต่างๆของ Blockchain Network ที่ต้องการได้ เช่น การยืนยันธุรกรรม 10-15 นาทีตามกฎของ Bitcoin แต่กลับกันถ้า Private Blockchain ของเราเองเราสามารถออกแบบให้การ Confirm ธุรกรรมแล้วเสร็จภายใน 2 วินาทีได้

ข้อเสีย องค์กรต้องลงทุนในการสร้างระบบ Infrastructure ขึ้นมาให้รองรับการทำงานภายนองค์กร ซึ่งมีความท้าทายในการดูแลรักษา และเม็ดเงินที่องค์กรจะต้องลงทุน

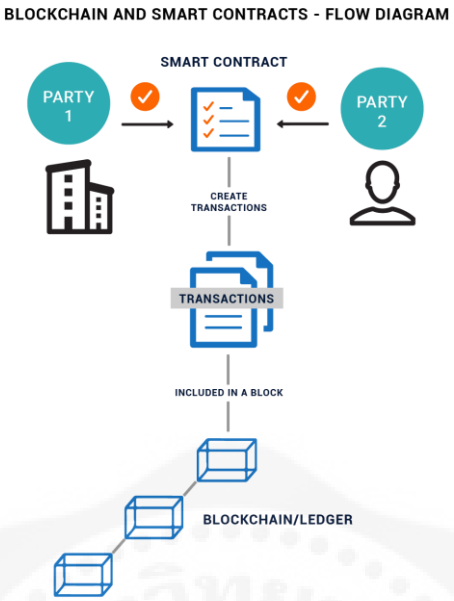
3. Consortium Blockchain เป็นบล็อกเชนที่รวมแนวคิดของบล็อกเชน public และ private เข้าด้วยกัน โดยนิยมใช้ในองค์กรต่างๆ ที่มีลักษณะธุรกิจเหมือนกันและต้องรับส่งแลกเปลี่ยนข้อมูลกันอยู่แล้ว เช่น ธนาคารใช้ในการแลกเปลี่ยนข้อมูลการโอนเงินกันภายในสมาคมธนาคารด้วยกัน และธนาคารที่จะเข้ามาร่วมในวงได้ต้องได้รับการอนุญาตจากตัวแทนเสียก่อนถึงจะมีสิทธิเข้าถึงการใช้งานร่วมกันได้

ข้อดี ธนาคารไม่ต้องกลัวว่าข้อมูลสำคัญขององค์กรและลูกค้าจะกลายเป็นข้อมูล Public และในเรื่องการลงทุน Infrastructure ก็ลดลงไม่เหมือนการสร้าง Private Blockchain ขึ้นมาเองภายในองค์กรที่มีต้นทุนไม่ต่างกับลงทุนทำระบบใหญ่ๆ ซักหนึ่งระบบ

ข้อเสีย ไม่คล่องตัวในการปรับปรุงเปลี่ยนแปลงเงื่อนไขการใช้งานต่างๆ เพราะอาจจะต้องผ่านมติเห็นชอบในสมาคมอย่างนี้เป็นต้น

2.3 ทฤษฎีเกี่ยวกับ Smart Contract [4]

Smart Contract คือ “สัญญาอัจฉริยะ” สำหรับคำว่า “สัญญา” ในที่นี้ อาจจะเป็นสัญญาการว่าจ้าง สัญญาการซื้อขาย หรือเอกสารของทางการ สิ่งนี่เริ่มจาก Nick Szabo เป็นผู้เสนอไอเดียว่า บล็อกเชนสามารถใช้ในการบันทึกข้อตกลงของสัญญาที่สามารถดำเนินการได้ด้วยตัวเอง ไม่จำเป็นต้องมีคนกลาง หรือใช้พนักงานในการมานั่งตรวจสอบเอกสาร ซึ่งจากรูปที่ 4 จะแสดงให้เห็นกระบวนการที่เกิดขึ้นเมื่อมีคนสองฝ่ายต้องการทำสัญญาจะให้คอมพิวเตอร์และโปรแกรมจัดการ และยังไม่สามารถทุจริตได้ เพราะทุกคนในบล็อกเชนจะเป็นพยานว่าสัญญานี้เกิดขึ้นและบรรลุจริง



รูปที่ 4 กระบวนการทำงานของ smart contract

https://www.pinterest.com/pin/463378249153508783/?lp=true

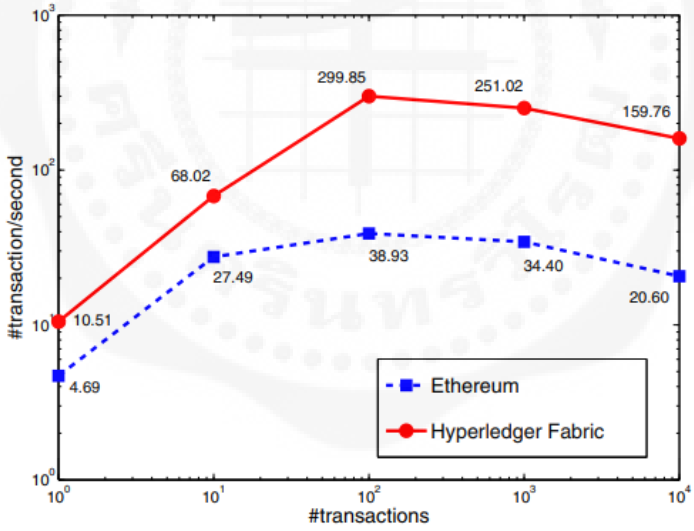
2.4 งานวิจัยที่เกี่ยวข้อง

- **Using blockchain to make land registry more reliable in India [5]**

ทดสอบแนวคิดที่ว่า ใช้ประโยชน์จากโครงสร้างการทำงานของ Ethereum โดยมุ่งเน้นการทำ Smart Contract สร้างสภาพแวดล้อมที่เป็นหนึ่งเดียวของสถานะการเป็นเจ้าของและประวัติของทรัพย์สิน ผู้ซื้อจะมั่นใจได้ว่าที่ดินที่ซื้อมานั้นเป็นที่ดินที่ถูกต้องและผู้ขายเป็นเจ้าของ ลดความเป็นไปได้ในการเกิดข้อพิพาทรวมทั้งค่าใช้จ่ายและเวลาที่เกี่ยวข้องกับการทำรายการใดๆ

- **Performance Analysis of Private Blockchain Platforms in Varying Workloads** [6]

บทความนี้นำเสนอการวิเคราะห์ประสิทธิภาพของแพลตฟอร์ม Private Blockchain ที่เป็นที่นิยม Hyperledger Fabric และ Ethereum (การใช้งานส่วนตัว) เพื่อประเมินประสิทธิภาพและข้อจำกัด ของแพลตฟอร์มเหล่านี้ เทคโนโลยีการทำธุรกรรมและการจัดการข้อมูลแบบกระจายศูนย์กลาง Blockchain ถือเป็นเทคโนโลยีที่จะมีผลกระทบเช่นเดียวกับอินเทอร์เน็ตในชีวิตของผู้คน หลายอุตสาหกรรมมีความสนใจในการนำ Blockchain มาใช้ในระบบไอทีของตน แต่ความสามารถยังเป็นที่กังวลที่ถูกกล่าวถึงอย่างมากในเทคโนโลยี Blockchain ปัจจุบัน ดังนั้นเป้าหมายของการวิเคราะห์ประสิทธิภาพเบื้องต้นนี้เป็นสองอย่าง ประการที่แรกมีการพัฒนาวิธีการประเมินแพลตฟอร์ม blockchain ประการที่สองผลการวิเคราะห์จะถูกนำเสนอเพื่อแจ้งให้ผู้ปฏิบัติงานทราบในการตัดสินใจเกี่ยวกับการยอมรับเทคโนโลยี blockchain ในระบบไอทีของตน ผลการทดลองขึ้นอยู่กับจำนวนธุรกรรมที่แตกต่างกันแสดงให้เห็นว่า Hyperledger Fabric มีประสิทธิภาพดีกว่า Ethereum จากรูปที่ 5 ในทุกตัวชี้วัดเวลาดำเนินการ , ความล่าช้าและปริมาณงาน (ต่อหน่วยเวลา) นอกจากนี้ทั้งสองแพลตฟอร์มยังคงไม่สามารถแข่งขันกับระบบฐานข้อมูลปัจจุบันในแง่ของประสิทธิภาพในสถานการณ์ภาระงานที่สูง



รูปที่ 5 กราฟแสดงถึง Throughput ของ Ethereum และ hyperledger fabric

- **Land Registry in Sweden** [13]

หน่วยงานด้านการจดทะเบียนที่ดินของประเทศสวีเดนอ้างว่า ก่อนที่จะมีการนำ private blockchain และ smart contracts มาใช้ การทำธุรกรรมอสังหาริมทรัพย์ในบางครั้งใช้เวลาหกถึงเก้าเดือน และคาดว่า เมื่อนำระบบนี้มาใช้จะช่วยลดขั้นตอนลงได้อย่างมาก ช่วยลดเวลาในการทำงานและลดความเป็นไปได้ที่จะเกิดข้อผิดพลาดของมนุษย์ในหลายขั้นตอนการทำงานและเนื่องด้วยข้อดีของระบบนี้ทำให้ ธนาคาร SBAB ของสวีเดนและธนาคาร Landshypotek เข้ามามีบทบาทในโครงการนี้ด้วย และในอนาคต หน่วยงานด้านการจดทะเบียนที่ดินของประเทศสวีเดน คาดว่าจะรวมหน่วยงานภาครัฐเข้ามาในระบบมากขึ้น เนื่องจากจำนวนของหน่วยงานที่เกี่ยวข้องและกฎหมายสวีเดนต้องมีลายเซ็นรับรองสำเนาถูกต้องสำหรับธุรกรรมอสังหาริมทรัพย์ และเมื่อระบบได้รับการพัฒนาอย่างเต็มที่แล้วคาดว่า การทำธุรกรรมอสังหาริมทรัพย์นี้จะลดเวลาจากหลายเดือน เหลือเพียงไม่กี่วัน

จากที่ทางคณะผู้วิจัยได้ทำการศึกษามาพบว่าในปัจจุบันยังไม่มีงานใดที่นำ blockchain ไปใช้มาช่วยลดกระบวนการจดจำนองที่ดิน ที่อ้างอิงจากระบบเดิมจากกรมที่ดินส่วนมากมักนำ blockchain มาทำเป็นระบบใหม่โดยไม่คำนึงถึงระบบเดิม การให้สิทธิการถือครองโหนดโดยอิสระ ทางคณะผู้วิจัยออกแบบระบบโดยอิงจากระบบเดิม การกำหนดสิทธิการถือครองโหนดโดยยึดตามหลักกฎหมาย และมีที่ปรึกษาจากทากรมที่ดินค่อยประสานงานร่วมกันเพื่อความถูกต้อง

บทที่ 3
การดำเนินงานวิจัย

ในการดำเนินงานผู้วิจัยจะนำเสนอองค์ประกอบต่างๆของงานวิจัย เพื่อให้สามารถเข้าใจถึงแนวคิดและกระบวนการที่มีความสำคัญในการพัฒนา Blockchain มาประยุกต์ใช้กับการเก็บโฉนดที่ดินและกระบวนการทำสัญญาจำนอง แบ่งเป็นหัวข้อ ดังนี้

3.1 กรอบแนวคิดการวิจัย

จากการที่กลุ่มของผู้วิจัยได้ทำการศึกษาและค้นคว้าข้อมูลที่เกี่ยวข้องกับการดำเนินงานของกรมที่ดินและการใช้งานโฉนดที่ดินในปัจจุบัน เพื่อนำเทคโนโลยี Blockchain เข้ามาพัฒนากระบวนการต่างๆให้ความน่าเชื่อถือและมีประสิทธิภาพมากขึ้น ทางกลุ่มของผู้วิจัยจึงทำการศึกษาค้นคว้าองค์ความรู้ในด้าน Blockchain อย่างละเอียด เพื่อประยุกต์ใช้โดยที่ไม่ขัดกับกระบวนการทำงานที่เป็นอยู่ในปัจจุบัน

การกลุ่มผู้วิจัยได้นำเครื่องมือต่างๆ ที่เกี่ยวข้องกับ Blockchain มาช่วยอำนวยความสะดวกในการพัฒนา เช่น go etherem ใช้ในการสร้างเครือข่าย Private Blockchain , Etheruem Wallet เครื่องมือที่ใช้ในการ Deploy Smart Contract , Solidity ภาษาที่ใช้ในการพัฒนา Smart Contract , platooreum เครื่องมือที่ช่วยในการจำลองเครือข่าย Blockchain อย่างง่าย , Web3.js เครื่องมือที่ใช้เชื่อมต่อกับ Smart Contract

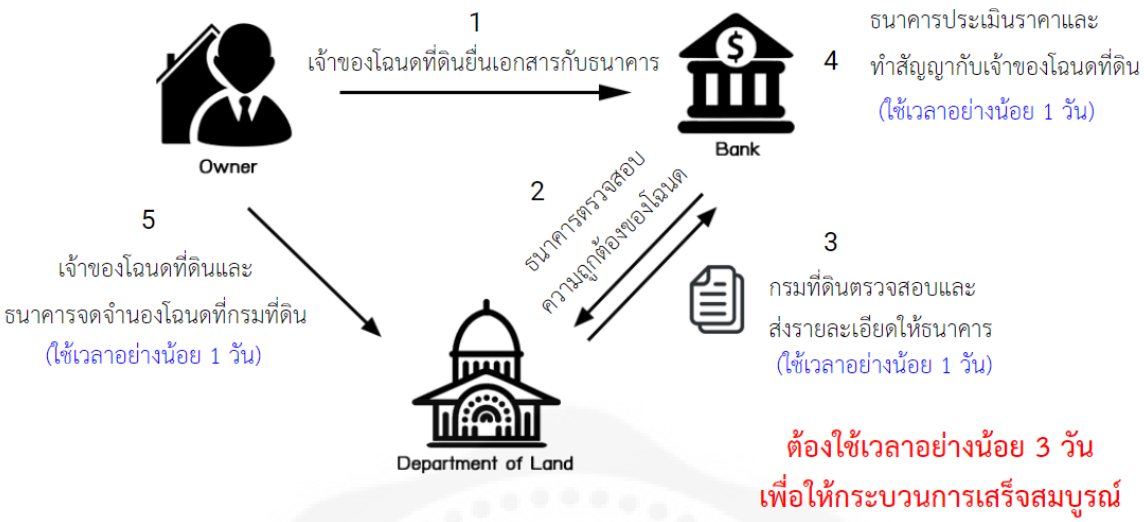
ดังนั้นกลุ่มของผู้วิจัยจึงได้นำองค์ความรู้ที่ศึกษาจากขั้นตอนการทำงานดังกล่าวมาทำการพัฒนา PoC ประยุกต์ใช้ Blockchain กับการเก็บโฉนดที่ดินและกระบวนการทำธุรกรรม

3.1.1 ความแตกต่างของกระบวนการจำนอง

กระบวนการจำนองปัจจุบัน

1. ยื่นเอกสารให้ธนาคารพิจารณา
2. ประเมินราคาหลักประกัน
   - ราคาประเมิน http://property.treasury.go.th/pvmwebsite/index.asp
   - ค้นหารูปแปลง http://dolwms.dol.go.th/tvwebp

   ธนาคารจะประเมินจาก
   - ราคาซื้อ-ขาย ของตลาด
   - ราคาประเมินกรมที่ดิน
   - ราคาทรัพย์ที่ธนาคารเคยปล่อยกู้ครั้งล่าสุด
3. ธนาคารอนุมัติ พร้อมแจ้งวงเงินและอัตราดอกเบี้ย
4. เซ็นสัญญากับธนาคาร + สัญญาจำนอง

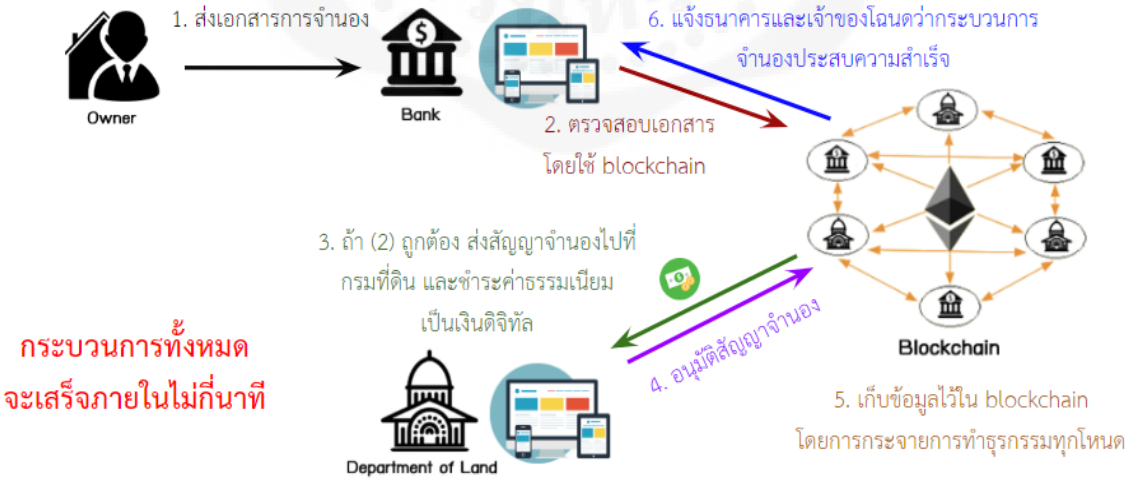


รูปที่ 6 แสดงถึงกระบวนการจำนองในปัจจุบัน ตามลำดับหมายเลข

กระบวนการหลังจากการนำ Blockchain เข้ามาใช้

1. ยื่นเอกสารให้ธนาคารพิจารณา
2. ประเมินราคาหลักประกัน
3. ธนาคารอนุมัติ พร้อมแจ้งวงเงินและอัตราดอกเบี้ย และ
   เซ็นสัญญาผ่าน smart contract

*** ไม่ต้องมีการนัดอีกครั้งเพื่อทำการเซ็นสัญญาที่กรมที่ดิน



รูปที่ 7 แสดงถึงกระบวนการจำนองโดยใช้ Blockchain ตามลำดับหมายเลข

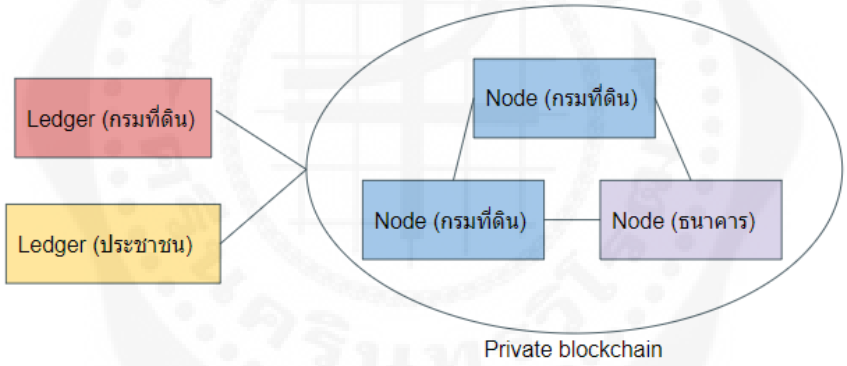
3.2 แนวคิดที่นำเสนอ

ผู้วิจัยทำการศึกษาและวิจัยข้อมูลที่เกี่ยวข้องกับการดำเนินงานของกรมที่ดินและการใช้โฉนดที่ดิน เพื่อนำเทคโนโลยี Blockchain มาใช้ในการพัฒนากระบวนการตรวจสอบโฉนดธุรกรรมของโฉนดที่ดินเช่น สัญญาจำนองให้มีความน่าเชื่อถือและมีประสิทธิภาพมากขึ้นซึ่งไม่ขัดแย้งกับกระบวนการทำงานในปัจจุบัน โดยแบ่งเป็น Layer ดังนี้

- Blockchain

พัฒนา blockchain ส่วนตัวโดยใช้ GETH เพื่อเตรียมสภาพแวดล้อม กรมที่ดินและผู้มีส่วนได้เสียทำหน้าที่เป็นผู้ควบคุมโหนดในระบบ และทำหน้าที่ยืนยันการทำงานในระบบ

ใช้บัญชีแยกประเภทเพื่อตรวจสอบตัวตนของบุคคลที่ใช้ระบบ รวมถึงคนที่มาใช้ด้วยเช่นกันการใช้กุญแจสาธารณะและกุญแจส่วนตัว



รูปที่ 8 โครงสร้าง Private Blockchain

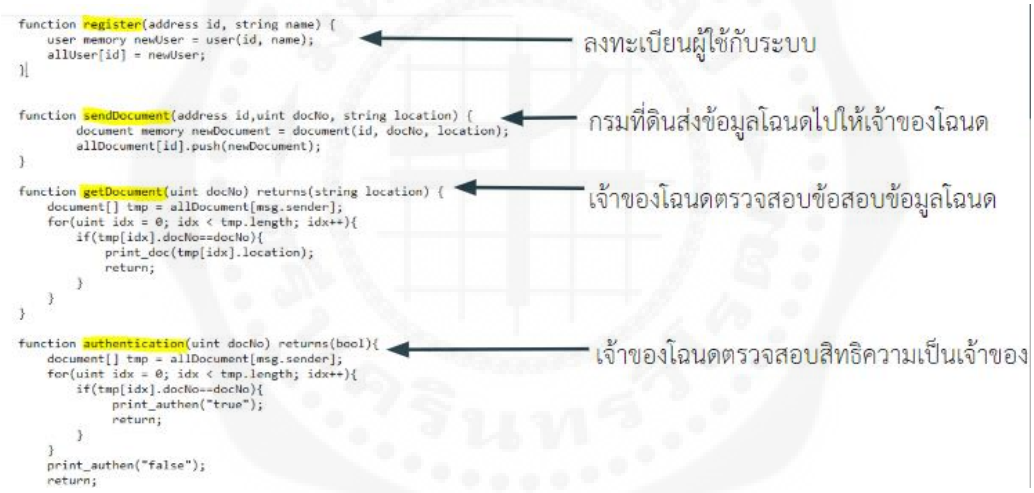
จากรูปที่ 8 แสดงถึงตัว Private Blockchain โดยมีกรมที่ดินและธนาคารช่วยกันถือครอง Node เพื่อสร้างเครือข่ายการเก็บข้อมูลแบบกระจาย โดยมี Ledger ให้กับกรมที่ดินและประชาชนเพื่อใช้งานในระบบ Blockchain

- Smart Contract

พัฒนา Contract โดยใช้ solidity 0.5.2 เพื่อจัดการโฉนด ข้อมูลโฉนดจะถูกรวบรวมดังนี้: {id, ส่วน, จำนวนของส่วน, เขต, หมายเลขเอกสาร, เขต, จังหวัด, ชื่อ, สัญชาติ, ขนาด} และธุรกรรมที่เกี่ยวข้องกับการกระทำที่ดิน เช่น

สัญญาจำนอง

1. กรมที่ดินลงทะเบียนข้อมูลโฉนดลงในระบบได้
2. ธนาคารทำธุรกรรมกับเจ้าของโฉนดที่ดิน
3. ธนาคารยกเลิกการทำธุรกรรมกับเจ้าของโฉนด
4. ผู้คนต้องการข้อมูลเกี่ยวกับที่ดินของตนเองเพื่อตรวจสอบ
5. คนเรียกข้อมูลธุรกรรมของโฉนดที่ดินของตนเองเพื่อตรวจสอบ

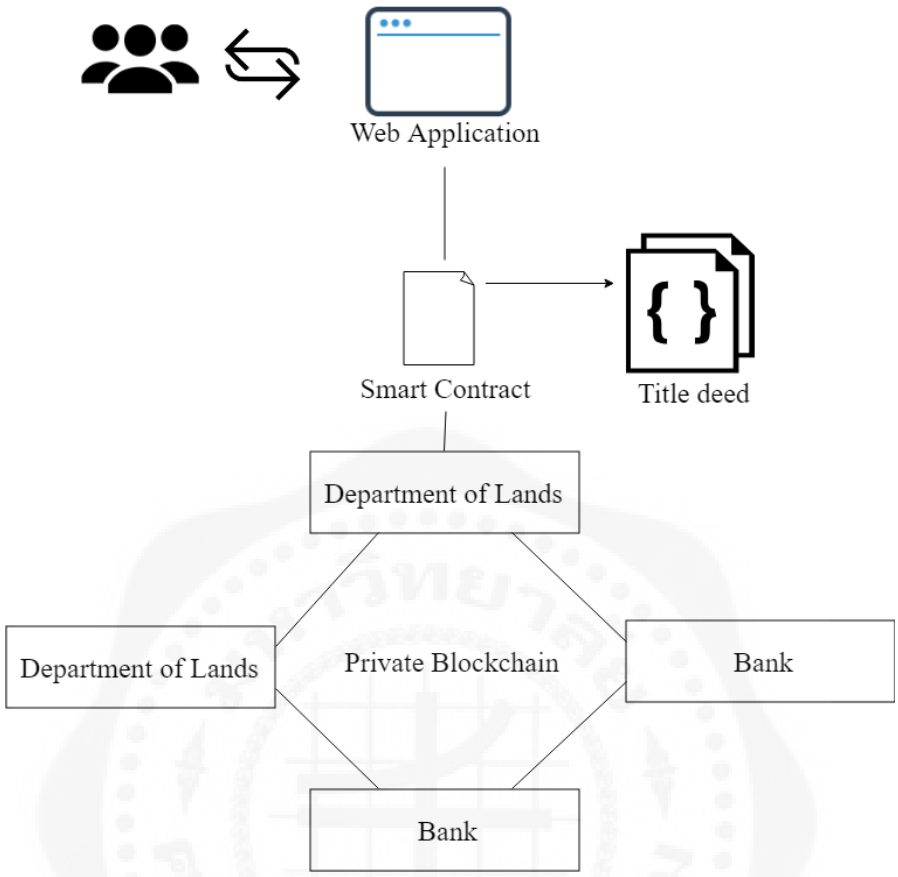


รูปที่ 9 solidity ที่ใช้ในทดสอบ Smart Contract

จากรูปที่ 9 แสดงถึงตัวอย่างโค้ดที่เขียนโดย solidity ที่เขียนเป็น fucntion ต่างๆ เพื่อใช้ในการดำเนินการกับข้อมูลโฉนดที่ดินและสัญญาจำนอง

- Web Application

พัฒนาเว็บแอปพลิเคชั่นเป็นอินเทอร์เฟซสำหรับกรมที่ดินธนาคารและสาธารณะเพื่อเข้าถึงเครือข่ายบล็อกเชนเพื่อทำกิจกรรมต่าง ๆ กับโฉนดที่ดิน
ซึ่งจาก Layer ดังกล่าว จะทำงานร่วมกัน ดังรูปที่ 10

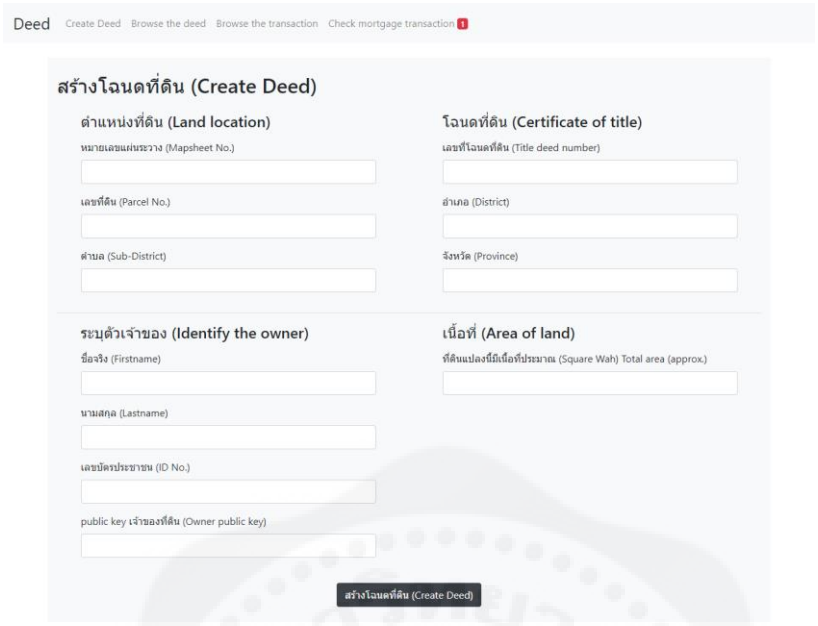


รูปที่ 10 แสดงถึงภาพรวมโครงสร้างของระบบ

3.3 รายละเอียดการทำงานของฟังก์ชั่น

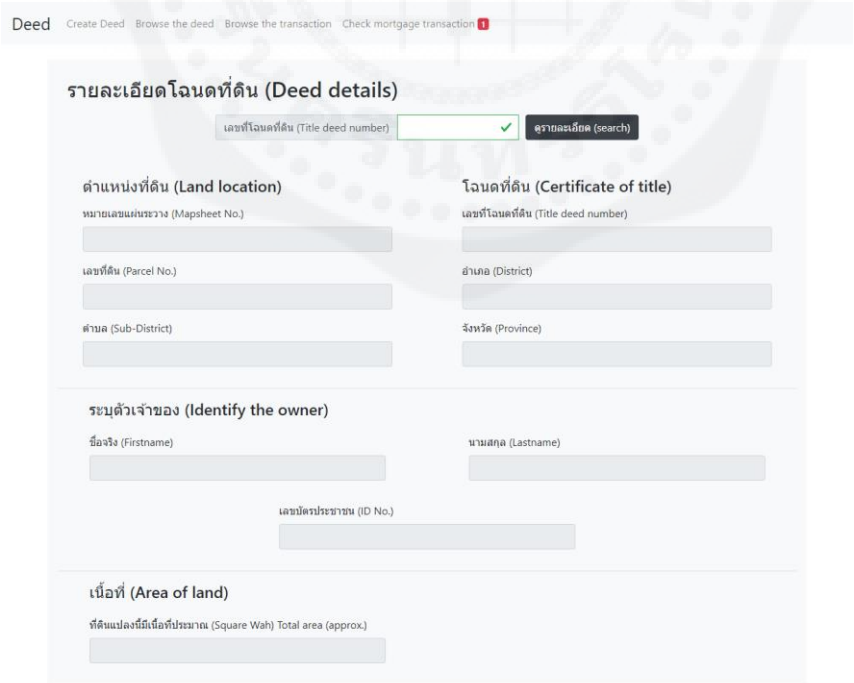
ในการทำงานในแต่ละฟังก์ชั่นผู้ใช้จะแตกต่างกันไปขึ้นอยู่กับจุดประสงค์ของการใช้งานซึ่งมีดังนี้

1. **sendDocument** เป็นฟังก์ชั่นสร้างเอกสารอิเล็กทรอนิกส์ของโฉนดที่ดินให้กับประชาชน
   - ผู้ใช้งาน : กรมที่ดิน
   - input : ตำแหน่งที่ดิน(ระวาง, เลขที่ดิน, ตำบล), โฉนดที่ดิน(เลขที่โฉนด, อำเภอ, จังหวัด), ระบุตัวเจ้าของโฉนดที่ดิน(public key เจ้าของโฉนดที่ดิน,ชื่อเจ้าของโฉนดที่ดิน, เลขบัตรประชาชน), ขนาดที่ดิน
   - authentication : account ของกรมที่ดิน



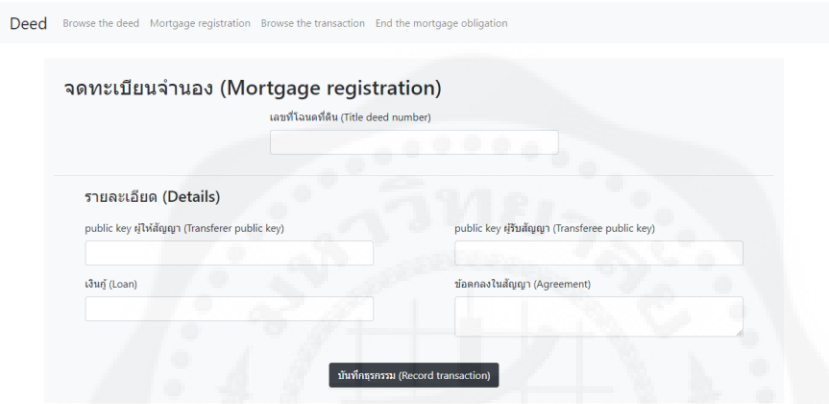
รูปที่ 11 sendDocument หรือ สร้างโฉนดที่ดิน

2. **getDocument** เป็นฟังก์ชั่นแสดงรายละเอียดของโฉนดที่ดินที่มีในครอบครอง
   - ผู้ใช้งาน : เจ้าของโฉนดที่ดิน
   - input : เลขที่โฉนด
   - authentication : account ของเจ้าของโฉนดที่ดิน



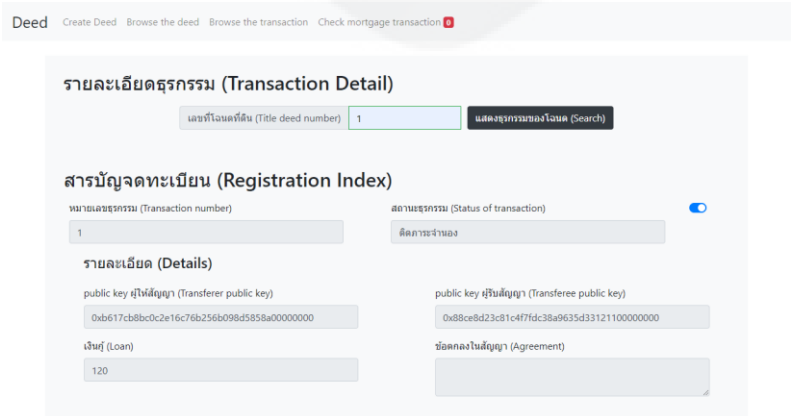
รูปที่ 12 getDocument หรือ รายละเอียดโฉนดที่ดิน

3. saveTransaction เป็นฟังก์ชั่นสร้างธุรกรรมสัญญาจำนองระหว่างธนาคารและเจ้าของโฉนดที่ดิน
    - ผู้ใช้งาน : ธนาคาร
    - input : เลขที่โฉนดที่ดิน, public key ของเจ้าของโฉนดที่ดิน , public key ของธนาคาร, จำนวนเงินในการจำนอง, เงื่อนไขการจำนอง
    - authentication : account ของเจ้าของโฉนดที่ดิน และ ธนาคาร



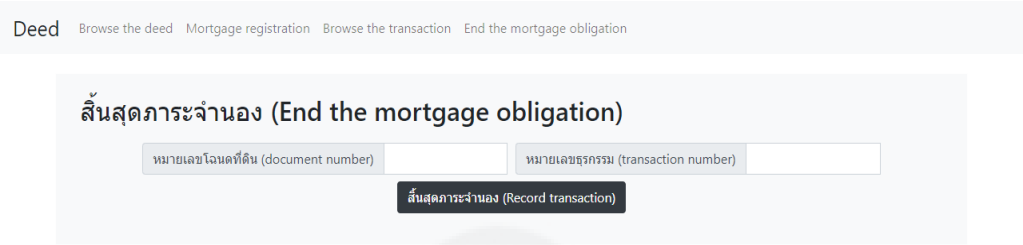
รูปที่ 13 saveTransaction หรือ จดทะเบียนจำนอง

4. getTransaction เป็นฟังก์ชั่นแสดงรายการธุรกรรมที่เกิดขึ้นกับโฉนดที่ดิน
    - ผู้ใช้งาน : เจ้าของโฉนดที่ดิน
    - input : เลขที่โฉนด
    - authentication : account ของเจ้าของโฉนดที่ดิน



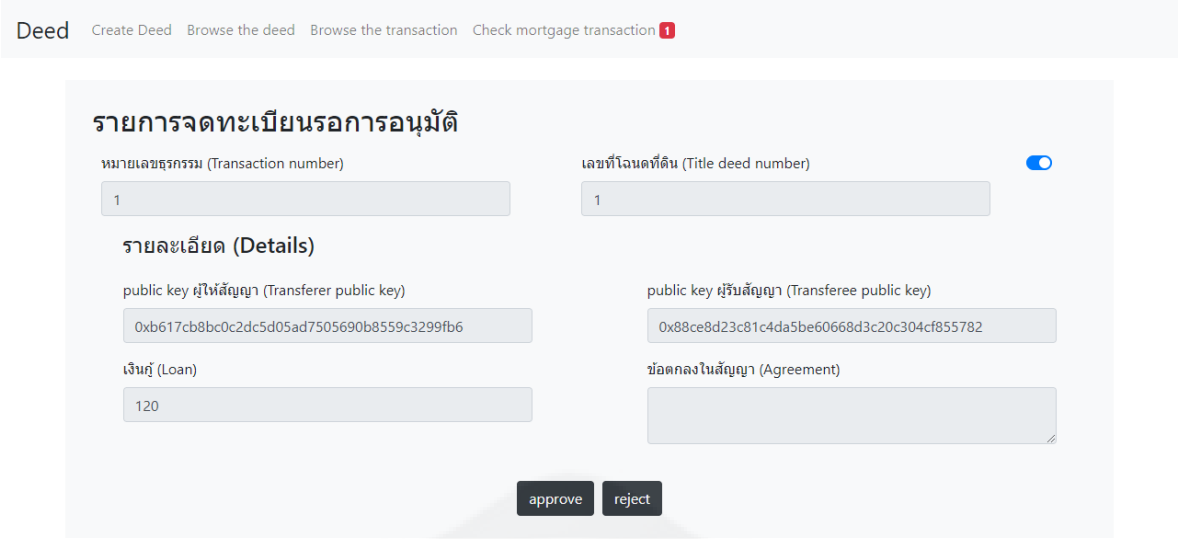
รูปที่ 14 gettracsaction หรือ รายละเอียดธุรกรรม

5. **endTransaction** เป็นฟังก์ชั่นทำธุรกรรมสิ้นสุดการจำนอง
   - ผู้ใช้งาน : ธนาคาร
   - input : เลขที่โฉนดที่ดิน, หมายเลขธุรกรรม
   - authentication : account ของธนาคาร



รูปที่ 15 endtransaction หรือ สิ้นสุดการจำนอง

6. **getWaitingList** เป็นฟังก์ชั่นแสดงรายการธุรกรรมสัญญาจำนอง
   - ผู้ใช้งาน : กรมที่ดิน
   - input : -
   - authentication : account ของกรมที่ดิน
7. **approveTransaction** เป็นฟังก์ชั่นอนุมัติธุรกรรมสัญญาจำนอง
   - ผู้ใช้งาน : กรมที่ดิน
   - input : index, เลขที่โฉนดที่ดิน, public key ของเจ้าของโฉนดที่ดิน ,public key ของธนาคาร, จำนวนเงินในการจำนอง,เงื่อนไขการจำนอง
   - authentication : account ของกรมที่ดิน
8. **rejectTransaction** เป็นฟังก์ชั่นไม่อนุมัติธุรกรรมสัญญาจำนอง
   - ผู้ใช้งาน : กรมที่ดิน
   - input : index
   - authentication : account ของกรมที่ดิน



รูปที่ 16 แสดงรายการธุรกรรมสัญญาจำนองที่รอการอนุมัติจากกรมที่ดิน

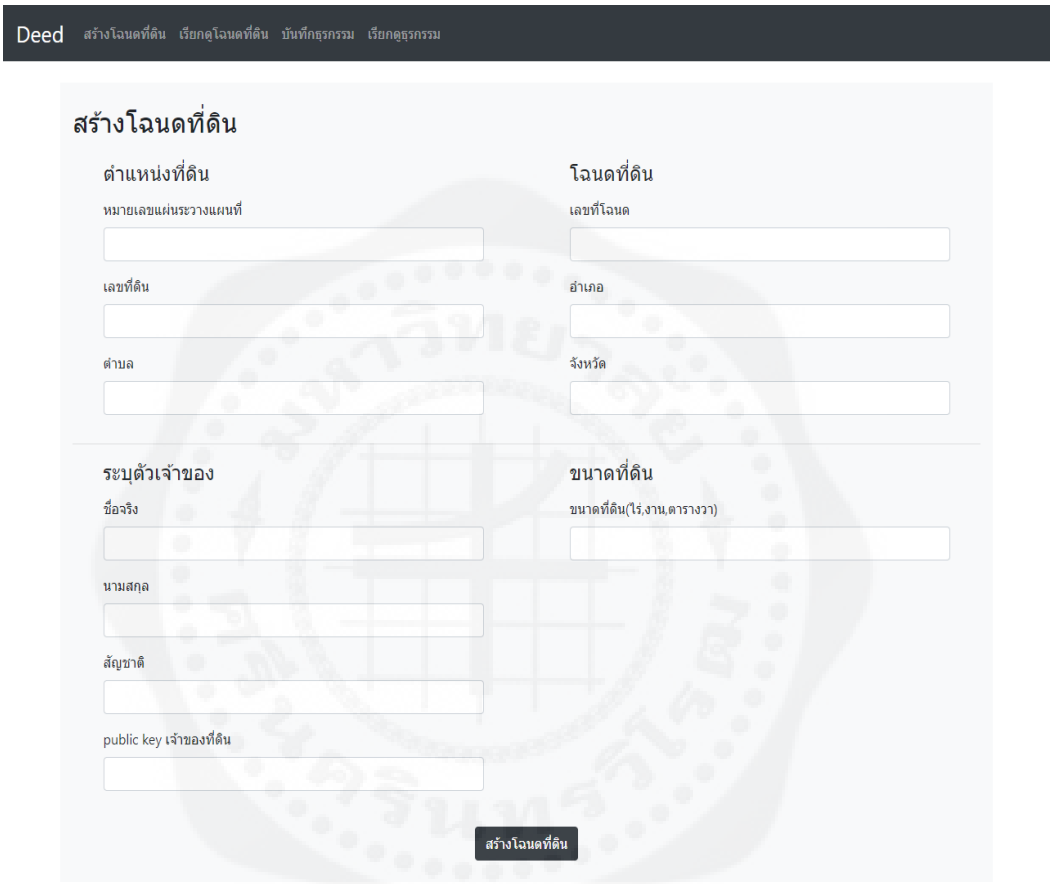
3.4 การทดสอบซอฟต์แวร์

ในการทดสอบระบบที่สร้างขึ้นมีความจำเป็นที่ต้องตั้งค่าสภาพแวดล้อมให้เหมาะสมต่อการทดสอบระบบที่สร้างขึ้นโดยจะมีการใช้ framwork ที่ช่วยในการทดสอบดังนี้

1. **ganache** เป็น framework ที่ใช้ในการตั้งค่าและเปิดใช้งาน node ของ ethereum โดยการตั้งค่า node ethereum จะสร้าง 10 บัญชีแต่ละบัญชีมี 100 eth , ตั้งค่าเวลายืนยัน transaction ให้เร็วขึ้น etc. โดยรวมเปิดการตั้งค่าให้ node ethereum เหมาะสมกับการทดสอบระบบในหลายๆส่วน
2. **truffle** เป็น framework ที่ใช้ในการ deploy smart contract ไปที่ ethereum node ซึ่งสามารถเพิ่มความสะดวกสบายในการ deploy smart contract ในจำนวนมากๆ และสามารถทดสอบฟังก์ชั่นใน smart contract ที่ truffle framework
3. **web3** เป็น javascript library ที่ใช้ในการติดต่อกับ smart contract บน ethereum node เพื่อมาแสดงข้อมูลบนหน้าเว็ป
4. **Metamask** เป็นส่วนเสริมสำหรับการเข้าถึงแอปพลิเคชันแบบกระจาย Ethereum หรือ "Dapps" ในเบราว์เซอร์

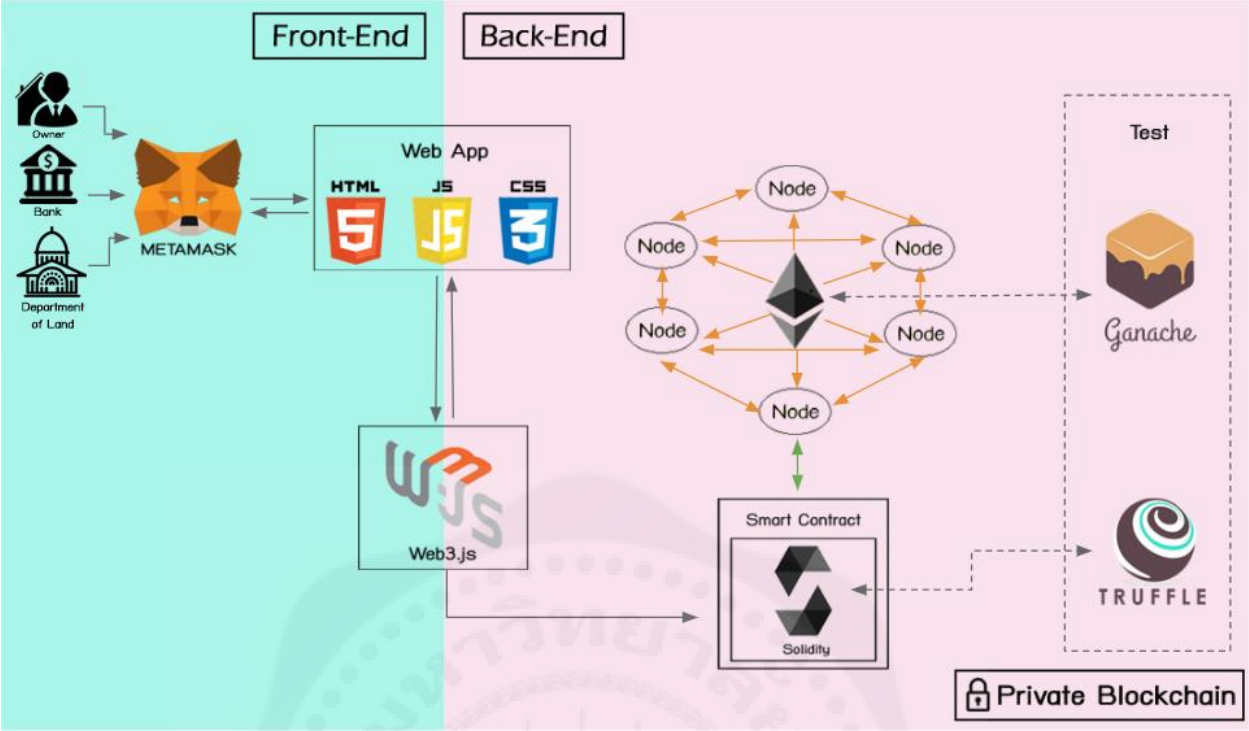
ต่อมาพัฒนาในส่วนของเว็ปเพื่อที่จะทำการทดสอบแสดงในรูปที่ 17 ซึ่งการสร้างโฉนดที่ดินในเว็บจะมีการกรอกข้อมูลที่มีส่วนเชื่อมโยงไปยังแฟ้มข้อมูลของกรมที่ดิน และการทำสัญญาในเว็บจะต้องกรอกข้อมูลคล้ายกับเอกสารสัญญาที่มีอยู่ในปัจจุบัน โดยผู้จัดทำจองลองเป็นสัญญาประเภทการจำนอง

ซึ่งการทำงานหน้าในเว็บประกอบไปด้วยหลายฟังก์ชั่น โดยแต่ละฟังก์ชั่นจะถูกกำหนดสิทธิการใช้งานโดยแบ่งออกเป็น 3 ส่วนคือ กรมที่ดิน, ธนาคาร และประชาชน ยกตัวอย่างเป็น การจะสร้างสัญญาจำนองจะต้องมาที่ธนาคารเพื่อที่จะสัญญาจำเป็นต้องใช้ public key and private key ของ ธนาคารและเจ้าของโฉนด โดยการทดสอบจะเน้นไปที่สิทธิในการใช้งานฟังก์ชั่น, ความปลอดภัยและความถูกต้องของข้อมูลเพื่อหาช่องโหว่ระหว่างหน้าเว็ป และ smart contracts



รูปที่ 17 Demo webpage.เป็นเว็ปที่ใช้สร้างโฉนดที่ดิน,
สร้างสัญญา etc. ตามฟังก์ชั่นใน smart contracts

โดยการมีการออกโครงสร้างของระบบตามรูปที่ 18 คือในส่วนของ front-end ออกแบบโดยใช้ html, js, bootstrap css และ metamask ใช้ในการจัดเก็บ public key และ private key เพื่อกำหนดสิทธิในการเข้าใช้งานในส่วนต่างๆ โดยใช้ web3 เชื่อมต่อกับ back-end ซึ่งในส่วนของ blockchain ใช้ ganache ในการสร้างขึ้นและใช้ truffle ในการ deploy smart contract ซึ่งง่ายในการทดสอบระบบ



รูปที่ 18 โครงสร้างของระบบที่พัฒนาขึ้นเพื่อทดสอบระบบ

3.5 ระบบความปลอดภัย

เนื่องจากเนื้อหาของงานวิจัยเกี่ยวข้องกับข้อมูลความเป็นส่วนตัวของบุคคลและข้อมูลเอกสารที่ไม่สามารถเปิดเผยให้คนทั่วไปทราบได้ ระบบความปลอดภัยจึงเป็นส่วนสำคัญที่ช่วยในการเก็บรักษาข้อมูลที่เป็นความลับ โครงสร้าง blockchain ที่นำเสนอมีวิธีการตรวจสอบหลายขั้นตอน ดังนี้

1. **การยืนยันตัวตนของผู้ใช้งาน**

Public-key cryptography คือการเข้ารหัสโดยกุญแจที่ใช้เข้ารหัสจะแตกต่างกับกุญแจที่ใช้ถอดรหัส นั่นคือการเข้ารหัสและการถอดรหัสจำเป็นต้องใช้กุญแจเป็นคู่ โดยที่บุคคลที่จะเข้ารหัสข้อมูลจะได้รับ กุญแจสาธารณะ (public key) ในการเข้ารหัส ส่วนบุคคลที่สามารถถอดรหัสได้คือบุคคลที่มี กุญแจส่วนตัว (private key) เท่านั้น กล่าวคือใคร ๆ ก็สามารถเข้ารหัสได้เพราะทุกคนมีกุญแจสาธารณะ (public key) แต่จะมีคนเดียวเท่านั้นที่ถอดรหัสได้คือคนที่มีกุญแจส่วนตัว (private key) ซึ่งต้องถูกเก็บไว้อย่างรัดกุม ใน blockchain ใช้คุณสมบัติของ Public-key cryptography เพื่อใช้ในการยืนยันตัวตนของผู้ใช้งาน เปรียบการใส่กุญแจส่วนตัว (private key) เหมือนการเซ็นลายเซ็น ทำให้ระบบสามารถแยกแยะและตรวจสอบผู้ใช้งานที่เข้ามาใช้งานในระบบได้

2. การระบุสิทธิ์ผู้ใช้งาน Web Application

แบ่งสิทธิการเข้าถึง Web Application โดยทำให้ User Interface ของแต่ละสิทธิมีลักษณะตามบทบาทของแต่ละ User ที่สามารถทำได้ เพื่อเป็นการป้องกันไม่ให้ผู้ที่ไม่มีสิทธิ์เข้าถึงสามารถเข้าถึง Smart Contract

3. การระบุสิทธิ์ผู้ใช้งาน Smart Contract

ในการเรียกใช้งาน Contract จะมีการตรวจสอบ require ของ Function ให้มีเงื่อนไขครบถ้วน ถูกต้องสำหรับการเรียกใช้งาน จึงสามารถดำเนินการตามขั้นตอนของ Function นั้นๆได้ เราจึงประยุกต์ใช้ส่วนนี้ในการตั่งข้อจำกัดใจการเข้าถึง Function ให้กับผู้ที่มีสิทธ์เข้าถึงเท่านั้น และการปลดล็อค Ledger เพื่อให้สามารถเรียกใช้งาน Contract ได้ โดยใช้ private key ทำให้ผู้ที่สามารถยืนยันการเรียกใช้งานได้ มีเพียงเจ้าของ Ledger

3.6 การวัดประสิทธิภาพ

หลังจากการพัฒนาระบบ และทำการปรับปรุงระบบเป็นที่เรียบร้อย ทางผู้วิจัยต้องการวัดประสิทธิภาพเพื่อทราบถึงขีดความสามารถของระบบที่สร้างขึ้นผ่านการจำลองสภาพแวดล้อมโดย

ผู้วิจัยออกแบบการทดลองโดยให้ผู้ใช้ส่งคำข้อ $\boldsymbol{N}$ transactions โดยส่งแบบ $\boldsymbol{f}$ ไปที่บล็อกเชนแพลตฟอร์ม Ethereum แบบไม่รอการตอบกลับ (asynchronous) ที่ติดตั้งเป็นจำนวน $\boldsymbol{B}$ Node โดยจะดูว่าระบบสามารถสร้าง transaction ได้สำเร็จทั้งหมดในเวลาเท่าไหรเพื่อเปรียบเทียบกับ transaction ของงานที่มีในกรมที่ดินปัจจุบัน

$\boldsymbol{N} \in$ {1,10,100,1000,10000}

$\boldsymbol{f} \in$ {SendDocumant,GetDocument,saveTransaction,getTransaction}

$\boldsymbol{B} \in$ {1,2,3,4}

โดยอ้างอิงจากข้อมูลทางสถิติของทางกรมที่ดินว่ามีประสิทธิภาพเพียงพอต่อการใช้งาน

3.7 แผนการดำเนินงาน

ขั้นตอน	เดือน									
	2561					2562				
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.
1) การวางแผนโครงการ										
ประชุมเพื่อเลือกหัวข้อ	x									
ศึกษาข้อมูลศึกษาแนวคิดและทฤษฎี		x								
รวบรวมปัญหา และ ความต้องการจากกรมทีดิน		x	x							
2) การวิเคราะห์ระบบ										
วิเคราะห์ความเป็นไปได้ในการทำระบบ			x	x						
3) การออกแบบระบบ										
กำหนดวัตถุประสงค์และขอบเขตของระบบ				x	x					
4) การพัฒนาระบบ										
ทำการพัฒนาระบบ				x	x	x				
เสนอให้กรมทีดินตรวจสอบ						x				
ทำการทดสอบและปรับปรุงระบบ							x			
5) สรุปและประเมินผล										
ทำการทดสอบประสิทธิของระบบ							x	x		
จัดทำคู่มือ และเอกสารทีเกียวข้อง								x	x	
สรุปผลการดำเนินงาน									x	
นำเสนอระบบทีสมบูรณ์										x

3.8 อุปกรณ์และเครื่องมือที่ใช้ในการวิจัย

ฮาร์ดแวร์ (Hardware) :

- ระบบปฏิบัติการ: Windows 10 Education (64-bit)
- หน่วยประมวลผล: Intel Core i5-3470 @ 3.20 GHz
- หน่วยความจำ: RAM 8 GB
- กราฟิก: NVIDIA GeForce 605 (1 GB VRAM)

ซอฟต์แวร์ (software) :

- **Go etheruem[7]** เป็นเครื่องมือบรรทัดคำสั่งอเนกประสงค์ที่รันโหนด Ethereum เต็มรูปแบบ
- **Platoo ruem[8]** เป็น platform ช่วยในรันโหนด Ethereum อย่างง่าย
- **Etheruem wallet[9]** ใช้ในการจัดการ Smart contract และทดสอบฟังก์ชั่นต่างๆ
- **Web3.js[10]** ชุดของไลบารี่ หรือ Ethereum JavaScript API ที่ช่วยให้สามารถเชื่อมต่อ และตอบโต้กับ Ethereum Node โดยเชื่อมต่อผ่าน HTTP หรือ IPC
- **NodeJS[11]** Platform ที่เขียนด้วย javaScript สำหรับเป็น Web Server ที่มีการประมวลผลที่เร็วขึ้น และทำงานแบบ Cross Platform โดยจุดเด่นของ nodeJS มีดังนี้

    1. Compiler V8 คือ JavaScript จาก Google ที่ทำให้เพิ่มประสิทธิภาพในการประมวลผลเร็วขึ้น

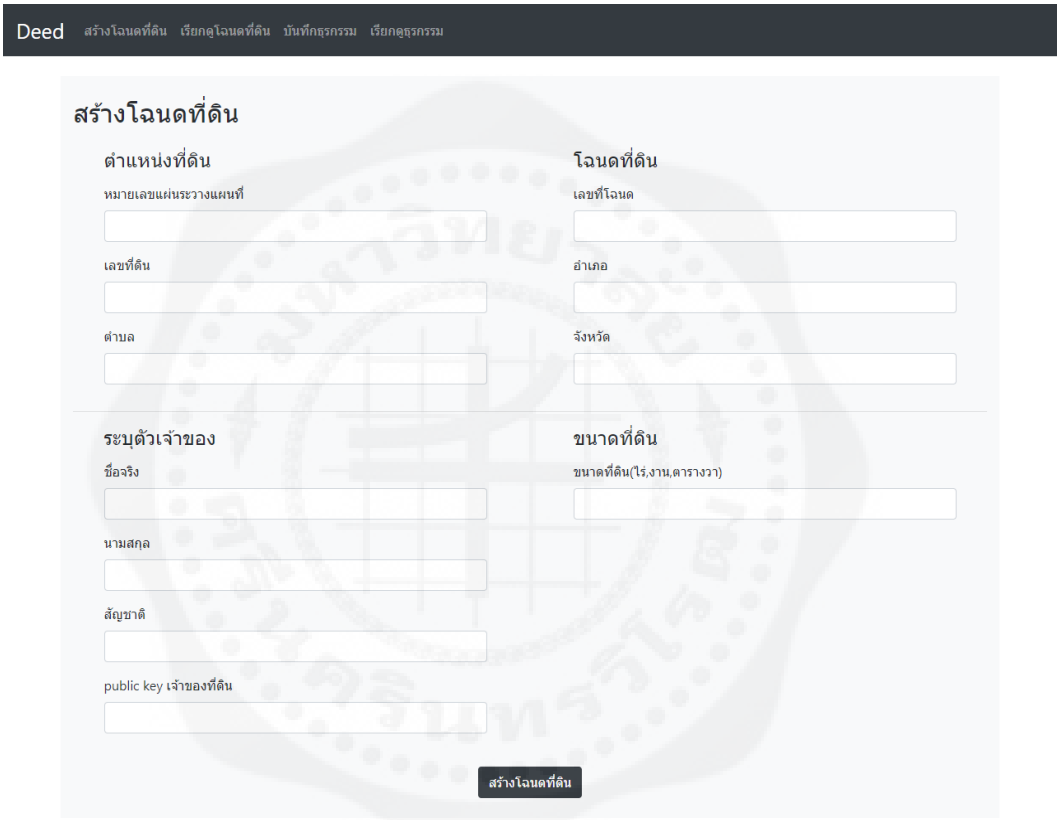
    2. ทำงานแบบ Cross Platform คือ เมื่อเขียนโค้ดหนึ่งขึ้นมาแล้ว สามารถนำไปรันได้ในระบบปฏิบัติการอื่นๆได้

    3. มีการทำงานแบบ Asynchronous เช่น C, PHP, .Net, Python หรือ ภาษาส่วนใหญ่มา เพราะว่าโดยปกติการทำงานของโค้ดที่เราเขียน จะทำงานจากบนลงล่างเสมอ และถ้าทำงานไม่เสร็จ ก็จะไม่ทำงานบรรทัดต่อไปเรื่อยๆ จนจบ แต่ node.js มีการทำงานเป็น Asynchronous คือ การทำงานบางอย่างไม่ต้องรอให้บรรทัดนั้นทำงานเสร็จ เช่น ส่งคำสั่งไป query ข้อมูลจาก database ไม่ต้องรอผล ก็ข้ามไปทำงานบรรทัดต่อไปเลย แล้วเมื่อการทำงานนั้นทำงานเสร็จก็ค่อยรอผลลัพท์กลับมา

บทที่ 4
ผลการดำเนินโครงการ

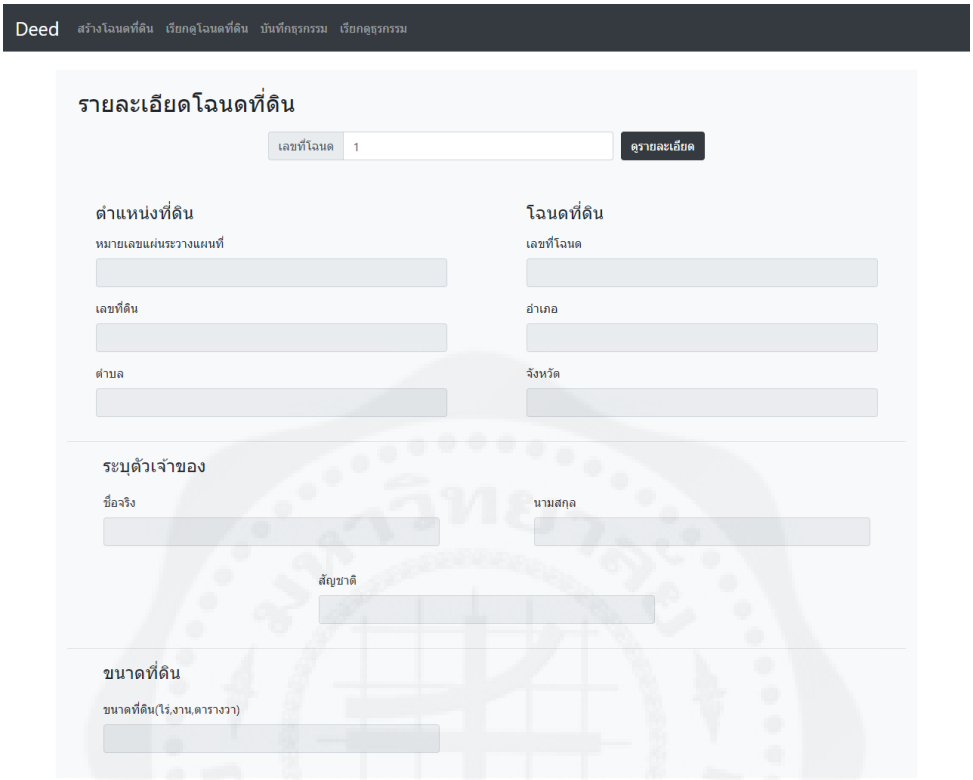
4.1 การทดสอบ Smart Contract

1.1 sendDocument สามารถสร้างโฉนดที่ดินให้กับผู้ใช้ที่เป็นเจ้าของโฉนด และเก็บข้อมูลลงใน Private Blockchain



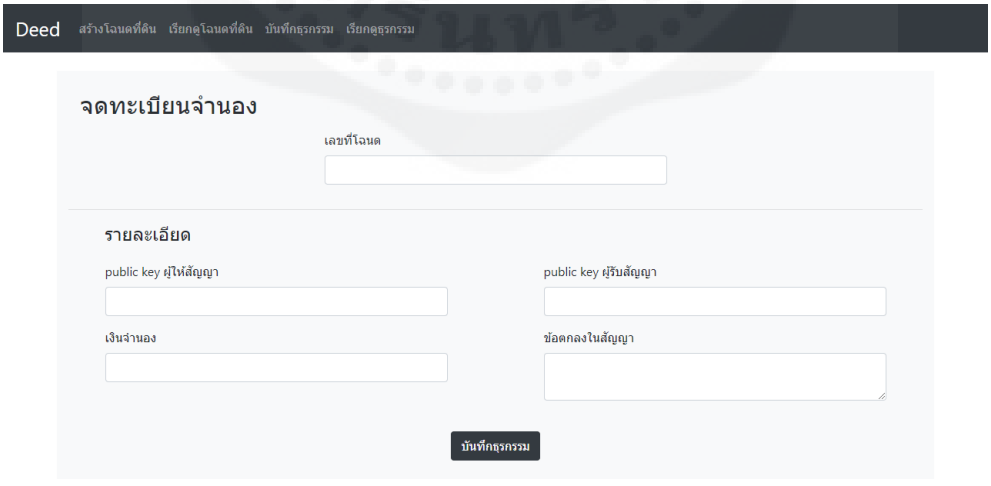
รูปที่ 19 ผลการทดสอบ sendDocument หรือ สร้างโฉนดที่ดิน

1.2 getDocument ผู้ใช้งานที่เป้นเจ้าขอโฉนดสามารถเรียกดูข้อมูลโฉนดที่ดินที่มีและสามารถเรียกดูได้แต่เพียงผู้เดียว



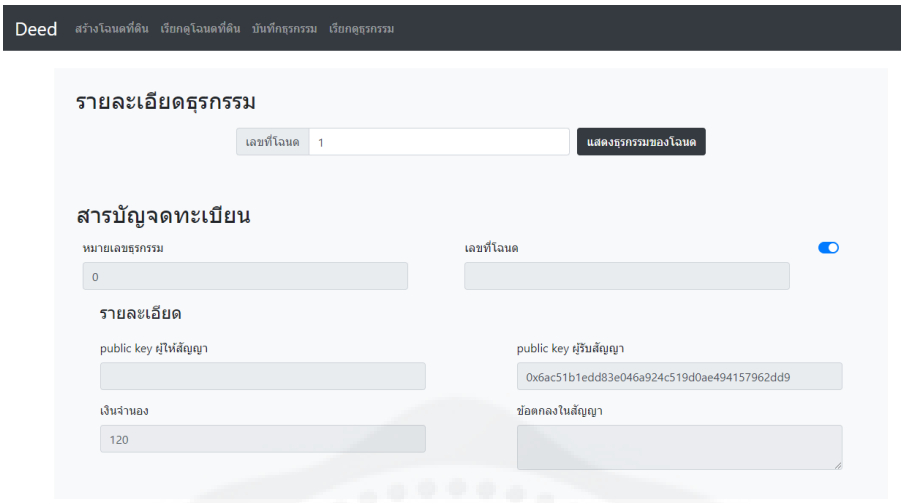
รูปที่ 20 ผลการทดสอบ getDocument หรือ รายละเอียดโฉนดที่ดิน

1.3 saveTransaction ผู้ใช้งานและธนาคาร สามารถทำธุรกรรมและเก็บบันทึกข้อมูลลงในโฉนดได้



รูปที่ 21 ผลการทดสอบ saveTransaction หรือ จดทะเบียนจำนอง

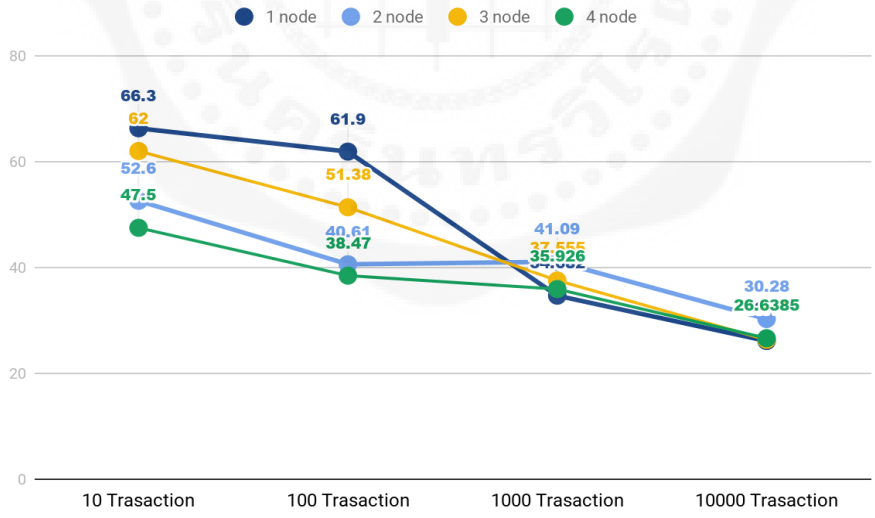
1.4 getTransaction ผู้ใช้งานที่เป็นเจ้าของโฉนดสามารถเรียกดูธุรกรรมที่เกิดขึ้นได้



รูปที่ 22 ผลการทดสอบ gettracsaction หรือ รายละเอียดธุรกรรม

4.2 ผลการทดสอบประสิทธิภาพ

จากการทดสอบโดยส่ง transaction เข้าไปในระบบ Blockchain และวัดเวลาที่ใช้ในการสำเร็จ transaction โดยได้ผลลัพดังภาพที่ 20



รูปที่ 23 กราฟแสดง throughput ของการทดสอบประสิทธิภาพ

บทที่ 5
สรุปผลการทดลอง

สรุปผลการดำเนินโครงการ

เทคโนโลยี Blockchain สามารถช่วยเพิ่มความโปร่งใสและทำธุรกรรมได้อย่างปลอดภัยยิ่งขึ้น ดังนั้นเราจึงนำประโยชน์นี้มาพัฒนาแอปพลิเคชั่นเป็น Proof-of-Concept (PoC) ที่สามารถนำไปใช้กับระบบการลงทะเบียนที่ดิน แม้ว่าจะมีข้อ จำกัด มากมายในการใช้ระบบที่เสนอในสภาพแวดล้อมจริง แต่งานวิจัยนี้ได้พัฒนาระบบให้ใกล้เคียงกับข้อมูลจริงและกระบวนการจริงเท่าที่จะทำได้ เพื่อประโยชน์และเป็นแนวทางสำหรับองค์กรที่จะสามารถปรับปรุงในอนาคต

โดยเราดำเนินการวิเคราะห์ประสิทธิภาพโดยมุ่งเน้นที่ปริมาณงานเป็นจำนวนธุรกรรมที่เสร็จสมบูรณ์ซึ่งดำเนินการต่อวินาที ผลลัพธ์แสดงว่าระบบที่เราเสนอเพียงพอต่อการจัดการธุรกรรม โดยปริมาณงานจากสถิติของกรมที่ดินอยู่ที่ 0.4 รายการต่อวินาที ซึ่งเพียงพอต่อธุรกรรมที่เกิดขึ้นกับกรมที่ดินในปัจจุบัน

ปัญหาและอุปสรรคในการดำเนินการ

- เวอชั่นของซอฟต์แวร์ที่ใช้พัฒนามีการอัพเดทอยู่บ่อยครั่ง ทำให้ระบบที่พัฒนาต้องอัพเดทตาม จึงเกิดข้อผิดพลาดและควมล่าช้า
- กระบวนการของระบบเกิดความซับซ้อนเนื่องจากต้องยึดตามกระบวนการจริงของกรมที่ดิน ที่ติดข้อกฎหมาย

ข้อเสนอแนะ

ในการพัฒนาระบบยังมีข้อจำกัดหลายอย่าง เช่น ผู้ใช้งานระบบ blockchain มี public key และ private key ของตัวเอง 1 โฉนดมีเจ้าของแค่ 1 คน ใช้ Smart contract ในการทำธุรกรรมสัญญาจำนองเท่านั้น และ จำนวน node ซึ่งสามารถเพิ่มขอบเขตการทำงานได้

- เพิ่มจำนวน node ในการทำสอบ
- ยกตัวอย่างธุรกรรมอื่นๆที่เกิดขึ้นกับโฉนดที่ดิน
- เพิ่มความปลอดภัยของข้อมูลโฉนดที่ดินที่จัดเก็บลง blockchain

แหล่งอ้างอิง

[1] nattaphon. (2560).ลายเซ็นอิเล็กทรอนิกส์ Digital Signature คืออะไร. สืบค้นเมื่อ 17 พ.ย. 2561. https://www.bcircle.co.th/2017/09/30/digital-signature/

[2] Tanya Sattaya-aphitan. (2561). ทบทวนความรู้เรื่อง Blockchain. สืบค้นเมื่อ 19 พ.ย. 2561.https://medium.com/@drtan/ทบทวนความรู้เรื่อง-blockchain-3691dded54a3

[3] Blockchain.Fish Team. (2559). Blockchain ประเภทไหนเหมาะกับองค์กรของคุณ. สืบค้นเมื่อ 19 พ.ย. 2561. https://blockchain.fish/องค์กรไหนควรใช้-blockchain-ประเภ/

[4] Earth. (2560). Smart Contract คืออะไร: เมื่อบล็อกเชนไม่ได้มีดีแค่บิทคอยน์. สืบค้นเมื่อ 19 พ.ย. 2561. https://siamblockchain.com/2017/06/08/smart-contract-คืออะไร

[5] Alexandru Oprunenco ; Chami Akmeemana (2561). USING BLOCKCHAIN TO MAKE LAND REGISTRY MORE RELIABLE IN INDIA. สืบค้นเมื่อ 10 ก.ย. 2561. http://www.undp.org/content/undp/en/home/blog/2018/Using-blockchain-to-make-land-registry-more-reliable-in-India.html

[6] Suporn Pongnumkul ; Chaiyaphum Siripanpornchana ; Suttipong Thajchayapong. (2560). Performance Analysis of Private Blockchain Platforms in Varying Workloads. สืบค้นเมื่อ 10 ก.ย. 2561. https://ieeexplore.ieee.org/document/8038517

[7] seandotau (2561). WHAT IS GETH?. สืบค้นเมื่อ 19 พ.ย 2561 http://www.talkcrypto.org/blog/2018/01/23/what-is-geth/

[8] Blockchain.Fish Team. (2560). เครื่องมือที่จะช่วยให้องค์กรของท่านมี Blockchain ใช้งานได้ใน 1 คลิก สืบค้นเมื่อ 19 พ.ย. 2561 https://blockchain.fish/platooreum-private-blockchain/

[9] Alex Van de Sande. (2559) .etheruem wallet-smart contract สืบค้นเมื่อ 19 พ.ย 2561 https://blog.ethereum.org/2015/12/03/how-to-build-your-own-cryptocurrency/

[10] web3.js - Ethereum JavaScript API. สืบค้นเมื่อ 19 พ.ย. 2561. https://web3js.readthedocs.io/en/1.0/

[11] อมรเดช คีรีพัฒนานนท์. (2559). node.js คืออะไร คือ programming language ที่เขียนด้วย JavaScript. สืบค้น เมื่อ 19 พ.ย. 2561. https://beyourcyber.com/2016/no de-js-is-programming-language-by-javascript/

[12] Crosby, M., Nachiappan, Pattanayak, P., Verma, S., & Kalyanaraman, V. (n.d.). (2560). BlockChain Technology. สืบค้นเมื่อ 19 พ.ย. 2561. http://scet.berkeley.edu/wpcontent/uploads/BlockchainPaper.pdf

[13] machetemag team. (2560). Land Registry in Sweden. สืบค้นเมื่อ 27 พ.ย. 2561. http://blockchain.ma chetemag.com/sweden/land-registry-sweden/

ภาคผนวก ก.

การติดตั้งซอฟต์แวร์

1. Node JS

   ดาวโหลดตัวติดตั้ง Node JS รุ่น LTS จาก https://nodejs.org/en/download/

   หลังจากดาวโหลดเสร็จสิ้นทำการ install ปกติ และ ตั้งค่า path environment

2. ganache

   เปิดโปรแกรม Command Prompt หรือ Powershell

   รันคำสั่ง npm install -g ganache-cli

3. truffle

   เปิดโปรแกรม Command Prompt หรือ Powershell

   รันคำสั่ง npm install -g truffle@4.0.4

ภาคผนวก ข.

โค้ดที่ใช้งาน

ชื่อไฟล์ : app.ja

ภาษาที่ใช้ : JavaScript

คำอธิบาย : ติดต่อกับ smart contract เพื่อรับส่งข้อมูล

```
App = {
  web3Provider: null,
  contracts: {},
  account: '0x',

  init: function() {
    return App.initWeb3();
  },
  //เรียกใช้ object ของ web3
  initWeb3: function() {
    // TODO: refactor conditional
    if (typeof web3 !== 'undefined') {
      // If a web3 instance is already provided by Meta Mask.
      App.web3Provider = web3.currentProvider;
      web3 = new Web3(web3.currentProvider);
    } else {
      // Specify default instance if no web3 instance provided
      App.web3Provider = new Web3.providers.HttpProvider('http://localhost:7545');
      web3 = new Web3(App.web3Provider);
    }
    return App.initContract();
```

```
},
//สร้าง object ที่ใช้เรียก smart contract 2 contract
initContract: function() {
  $.getJSON("Test_FullData.json", function(data) {
    // Instantiate a new truffle contract from the artifact
    App.contracts.Test_FullData = TruffleContract(data);
    // Connect provider to interact with contract
    App.contracts.Test_FullData.setProvider(App.web3Provider);

    $.getJSON("TokenERC20.json", function(data) {
      // Instantiate a new truffle contract from the artifact
      App.contracts.dolToken = TruffleContract(data);
      // Connect provider to interact with contract
      App.contracts.dolToken.setProvider(App.web3Provider);

      App.render();

    });

  });
},
//rander หน้า html ในบางส่วนที่มีการเรียกข้อมูลจาก smart contract
render: function() {
  $("#nav-placeholder").load("nav.html");
  var accountInterval = setInterval(function() {
    web3.eth.getCoinbase(function(err, account) {
      if (err === null) {
        if (App.account != account) {
```

```
App.account = account;
$('#PFaccount').val(App.account);
App.getBalance()
if (App.account == 0x2B8F79184073919Db5b037eEf0a9Cf3f0B20910F) {
  App.getWaitingList();
  $('#createLi').show();
  $('#getDeedLi').show();
  $('#saveTransactionLi').hide();
  $('#getTransactionLi').show();
  $('#endTransaction').hide();
  $('#approveTransaction').show();
  $('#transferMoney').show();
}else if (App.account == 0x723C6DB5E4A01B647Eb61e1094032eea969D9626) {
  $('#insertValueApprove').empty();
  $('#createLi').hide();
  $('#getDeedLi').show();
  $('#saveTransactionLi').show();
  $('#getTransactionLi').show();
  $('#endTransaction').show();
  $('#approveTransaction').hide();
  $('#transferMoney').hide();
}else {
  $('#insertValueApprove').empty();
  $('#createLi').hide();
  $('#getDeedLi').show();
  $('#saveTransactionLi').hide();
  $('#getTransactionLi').show();
  $('#endTransaction').hide();
```

```
            $('#approveTransaction').hide();
            $('#transferMoney').hide();
          }
        }
      }
    });
  }, 100);
},
//เรียกดู public key ของผู้ใช้งาน
showAccount: function(){
  $('#username').val(App.account);
},
//เรียกดูเงิน DOL ในบัญชีนั้นๆ
getBalance: function(){
  App.contracts.dolToken.deployed().then(function(instance) {
    instance.balanceOf($('#PFaccount').val().toString()).then(function(result) {
      var x = result.c[0] / 10 ** (result.c[0].toString().length - 1) ;
      var y = x * 10 ** result.e ;
      var z = y / (10 ** 18) ;
      $('#PFbalance').val(z + " DOL");
    });
  });
},
//โอนเงิน DOL
transferMoney: function(){
  var transferee =  $('#TMtransferee').val();
  var money =  $('#TMmoney').val() * (10 ** 18);
  App.contracts.dolToken.deployed().then(function(instance) {
```

```
      return instance.transfer(transferee, money, { from: App.account });
      }).then(function (result) {
        $("#fail").hide();
        $("#complete").show();
        $("#transactionStatus").modal();
      }).catch(function(err){
        console.log(err);
        $("#complete").hide();
        $("#fail").show();
        $("#transactionStatus").modal();
      });
  },
  //เรียกดูรายการจำนองที่รอการอนุมัติ
  getWaitingList: function(){
    $('#insertValueApprove').empty();
    App.contracts.Test_FullData.deployed().then(function(instance) {
      return instance.getWaitingList({ from: App.account });
    }).then(function (result) {
      $('#insertAlert').empty();
      $('#insertAlert').append(result.valueOf()[0].length);
      for (var i = 0; i < result.valueOf()[0].length; i++) {
        $('#insertValueApprove').append("<div class='row justify-content-start'> <div
class='form-group container col-md-5'> <label for='transactionNumber'>หมายเลขธุรกรรม
(Transaction number)</label> <input type='text' readonly class='form-control'
value='"+result.valueOf()[0][i]+"'> </div> <div class='form-group container col-md-5'>
<label for='documentNumber'>เลขที่โฉนดที่ดิน (Title deed number)</label> <input
type='text' readonly class='form-control' value='"+result.valueOf()[3][i]+"'></div> <div
class='form-group custom-switch col-md-1'> <input type='checkbox' class='custom-
```

control-input' id='customSwitch"+i+"'
onclick=\"customSwitchForm('customSwitch"+i+"');\"> <label class='custom-control-label'
for='customSwitch"+i+"'></label>   <div
class='row justify-content-start'> <h4>
รายละเอียด (Details)</h4>   <div
class='form-group container col-md-5'> <label for='nameGiver'>public key ผู้ให้สัญญา
(Transferer public key)</label> <input type='text' id='puGiver"+i+"' readonly class='form-
control' value='"+result.valueOf()[5][i]+"' ></input> <div class='form-group
container col-md-5'> <label for='nameReceiver'>public key ผู้รับสัญญา (Transferee public
key)</label> <input type='text' id='puReceiver"+i+"' readonly class='form-control'
value='"+result.valueOf()[1][i]+"'>
<label for='estateContract'>เงินกู้ (Loan)</label> <input type='text' readonly class='form-
control' value='"+result.valueOf()[2][i]+"'> <div class='form-group container col-md-
5'><label for=\"estateContract\">ค่าธรรมเนียม (Fee)</label><input type=\"text\"
class=\"form-control\" id=\"STfee\" readonly
value='"+result.valueOf()[4][i]+"'></input> <div class='form-group container col-
md-5'> <label for='estateRemain'>ข้อตกลงในสัญญา (Agreement)</label><label> 1.)
ข้าพเจ้าไม่ได้รับโอนแทนหรือเพื่อประโยชน์แก่บุคคลต่างด้าว</label><label> 2.)ราคาทรัพย์ที่
แสดงไว้ใน 4.) เป็นราคาที่แท้จริง</label><label> 3.)สิ่งปลูกสร้างในที่ดินแปลงนี้ที่มีอยู่แล้วและที่จะมี
ขึ้นในภายหน้าจำนองด้วยทั้งสิ้น</label><label> 4.)จำนองเพื่อเป็นประกันการกู้ยืมเงินซึ่งผู้จำนองได้กู้
จากผู้รับจำนอง</label><label> 5.)อัตราดอกเบี้ยร้อยละ "+result.valueOf()[6][i]+" ต่อปี และนำส่ง
ดอกเบี้ยปีละหนึ่งครั้ง</label>
<div class='row justify-content-
center'><button type='button' id='approveTracsaction' class='btn btn-dark'
onclick='App.approveTransaction("+i+","+result.valueOf()[3][i]+","+i+","+i+","+result.valueOf
()[2][i]+","+result.valueOf()[6][i]+","+result.valueOf()[4][i]+");'>approve</button> &nbs
p;<button type='button' id='rejectTracsaction' class='btn btn-dark'
onclick='App.rejectTransaction("+i+","+result.valueOf()[4][i]+");'>reject</button>
</div
>
")

```
        $("#" + "customSwitch" + i + "Form").hide();
      }
    }).catch(function(err){
      console.log(err);
    });
  },
  //สร้างโฉนดที่ดิน
  createDeed: function() {
    var section =  $('#Csection').val();
    var numberOfsection =  $('#CnumberOfsection').val();
    var district =  $('#Cdistrict').val();
    var documentNumber =  $('#CdocumentNumber').val();
    var county =  $('#Ccounty').val();
    var province =  $('#Cprovince').val();
    var name  =  $('#Cfirstname').val() + " " +  $('#Clastname').val(); ;
    var nationality =  $('#Cnationality').val();
    var ownerAddress =  $('#CownerAddress').val();
    var size =  $('#Csize').val();
    App.contracts.Test_FullData.deployed().then(function(instance) {
      return
instance.sendDocument(ownerAddress,section,numberOfsection,district,documentNumb
er,
        county,province,name,nationality,size, { from: App.account });
      }).then(function (result) {
        $("#fail").hide();
        $("#complete").show();
        $("#transactionStatus").modal();
      }).catch(function(err){
```

```
    console.log(err);
    $("#complete").hide();
    $("#fail").show();
    $("#transactionStatus").modal();
  });
},
//ดูรายละเอียดของโฉนดที่ดิน
getDeed: function() {
  var documentNumber =  $('#GdocumentNumber').val();
  App.contracts.Test_FullData.deployed().then(function(instance) {
    return instance.getDocument(documentNumber, { from: App.account });
  }).then(function (result) {
    $('#Gsection').val(result.valueOf()[0].toString());
    $('#GnumberOfsection').val(result.valueOf()[1].toString());
    $('#Gdistrict').val(result.valueOf()[2].toString());
    $('#GdocumentNumberR').val(result.valueOf()[3].toString());
    $('#Gcounty').val(result.valueOf()[4].toString());
    $('#Gprovince').val(result.valueOf()[5].toString());
    var tmpName = result.valueOf()[6].toString().split(" ");
    $('#Gfirstname').val(tmpName[0]);
    $('#Glastname').val(tmpName[1]);
    $('#Gnationality').val(result.valueOf()[7].toString());
    $('#Gsize').val(result.valueOf()[8].toString());
    $("#fail").hide();
    $("#complete").show();
    $("#transactionStatus").modal();
  }).catch(function(err){
    console.log(err);
```

```
        $("#complete").hide();
        $("#fail").show();
        $("#transactionStatus").modal();
      });
    },
    //สร้างสัญญาจำนอง
    createTransaction: function() {
      var documentNumber =  $('#STdocumentNumber').val();
      var fee =  $('#STfee').val();
      var interest =  $('#STinterest').val();
      var money =  $('#STmoney').val();
      var puGiver =  $('#STnameGiver').val();
      var puReceiver =  $('#STnameReceiver').val();

      var feeT = fee * (10 ** 18);

      App.contracts.dolToken.deployed().then(function(instance) {
        return instance.transfer("0x2B8F79184073919Db5b037eEf0a9Cf3f0B20910F", feeT, {
from: App.account });
        }).then(function (result) {
          App.contracts.Test_FullData.deployed().then(function(instance) {
            return
instance.saveTransaction(documentNumber,puGiver,puReceiver,money,interest,fee, {
from: App.account });
          }).then(function (result) {
            $("#fail").hide();
            $("#complete").show();
            $("#transactionStatus").modal();
```

```
          }).catch(function(err){
            console.log(err);
            $("#complete").hide();
            $("#fail").show();
            $("#transactionStatus").modal();
          });
        }).catch(function(err){
          console.log(err);
          $("#complete").hide();
          $("#fail").show();
          $("#transactionStatus").modal();
        });
    },
    //อนุมิติสัญญาจำนอง
    approveTransaction: function(index, documentNumber, puGiver, puReceiver, money,
interest, fee) {
      var giver = $('#puGiver'+puGiver+'').attr("value");
      var receiver =  $('#puReceiver'+puReceiver+'').attr("value");
      App.contracts.Test_FullData.deployed().then(function(instance) {
        return
instance.approveTransaction(index,documentNumber,giver,receiver,money,interest,fee, {
from: App.account });
      }).then(function (result) {
        $("#fail").hide();
        $("#complete").show();
        $("#transactionStatus").modal();
        App.getWaitingList();
      }).catch(function(err){
```

```
      console.log(err);
      $("#complete").hide();
      $("#fail").show();
      $("#transactionStatus").modal();
    });
},
//ไม่อนุมัติสัญญาจำนอง
rejectTransaction: function(index, fee){
  var receiver =  $('#puReceiver'+index+'').attr("value");
  var feeT = fee * (10 ** 18);
  App.contracts.dolToken.deployed().then(function(instance) {
    return instance.transfer(receiver, feeT, { from: App.account });
    }).then(function (result) {
      App.contracts.Test_FullData.deployed().then(function(instance) {
        return instance.rejectTransaction(index, { from: App.account });
      }).then(function (result) {
        $("#fail").hide();
        $("#complete").show();
        $("#transactionStatus").modal();
        App.getWaitingList();
      }).catch(function(err){
        console.log(err);
        $("#complete").hide();
        $("#fail").show();
        $("#transactionStatus").modal();
      });
    }).catch(function(err){
      console.log(err);
```

```
        $("#complete").hide();
        $("#fail").show();
        $("#transactionStatus").modal();
      });
  },
  //เรียกดูรายละเอียด transaction ของโฉนดที่ดิน
  getTransaction: function() {
    var documentNumber =  $('#GTdocumentNumber').val();
    var statusว
    App.contracts.Test_FullData.deployed().then(function(instance) {
      return instance.getTransaction(documentNumber, { from: App.account });
    }).then(function (result) {
      for (var i = 0; i < result.valueOf()[0].length; i++) {
        if(result.valueOf()[3][i] == false){
          status = "ติดภาระจำนอง" ;
        }else{
          status = "ไมติดภาระจำนอง" ;
        }
        $('#insertValue').append("<div class='row justify-content-start'> <div class='form-
group container col-md-5'> <label for='transactionNumber'>หมายเลขธุรกรรม (Transaction
number)</label> <input type='text' readonly class='form-control'
value='"+result.valueOf()[0][i]+"'> </div> <div class='form-group container col-md-5'>
<label for='documentNumber'>สถานะธุรกรรม (Status of transaction)</label> <input
type='text' readonly class='form-control' value='"+status+"'></div> <div class='form-
group custom-switch col-md-1'> <input type='checkbox' class='custom-control-input'
id='customSwitch"+i+"' onclick=\"customSwitchForm('customSwitch"+i+"');\"> <label
class='custom-control-label' for='customSwitch"+i+"'></label> </div> </div> <div
id='customSwitch"+i+"Form'> <div class='row justify-content-start'> <div class='form-
```

```
group container col-md-11'><h4>รายละเอียด (Details)</h4></div> </div> <div class='row
justify-content-start'> <div class='form-group container col-md-5'> <label
for='nameGiver'>public key ผู้ให้สัญญา (Transferer public key)</label> <input type='text'
readonly class='form-control' value='"+result.valueOf()[5][i]+"' > </div> <div class='form-
group container col-md-5'> <label for='nameReceiver'>public key ผู้รับสัญญา (Transferee
public key)</label> <input type='text' readonly class='form-control'
value='"+result.valueOf()[1][i]+"'> </div> <div class='form-group container col-md-5'>
<label for='estateContract'>เงินกู้ (Loan)</label> <input type='text' readonly class='form-
control'  value='"+result.valueOf()[2][i]+"'> </div> <div class='form-group container col-
md-5'><label for=\"estateContract\">ค่าธรรมเนียม (Fee)</label><input type=\"text\"
class=\"form-control\" id=\"STfee\" readonly
value='"+result.valueOf()[4][i]+"'></input></div>  <div class='form-group container col-
md-5'> <label for='estateRemain'>ข้อตกลงในสัญญา (Agreement)</label><label>    1.)
ข้าพเจ้าไม่ได้รับโอนแทนหรือเพื่อประโยชน์แก่บุคคลต่างด้าว</label><label>          2.)ราคาทรัพย์ที่
แสดงไว้ใน 4.) เป็นราคาที่แท้จริง</label><label> 3.)สิ่งปลูกสร้างในที่ดินแปลงนี้ที่มีอยู่แล้วและที่จะมี
ขึ้นในภายหน้าจำนองด้วยทั้งสิ้น</label><label>  4.)จำนองเพื่อเป็นประกันการกู้ยืมเงินซึ่งผู้จำนองได้กู้
จากผู้รับจำนอง</label><label> 5.)อัตราดอกเบี้ยร้อยละ "+result.valueOf()[6][i]+" ต่อปี และนำส่ง
ดอกเบี้ยปีละหนึ่งครั้ง</label> </div> </div><br>");
          $("#" + "customSwitch" + i + "Form").hide();
        }
        $("#fail").hide();
        $("#complete").show();
        $("#transactionStatus").modal();
      }).catch(function(err){
        console.log(err);
        $("#complete").hide();
        $("#fail").show();
        $("#transactionStatus").modal();
```

```
    });
  },
  //ยุติการจำนอง
  endTransaction: function() {
    var tranno =  $('#transactionNumber').val();
    var documentNumber = $('#deedNumber').val();
    App.contracts.Test_FullData.deployed().then(function(instance) {
      return instance.endTransaction(documentNumber,tranno, { from: App.account });
    }).then(function (result) {
      $("#fail").hide();
      $("#complete").show();
      $("#transactionStatus").modal();
    }).catch(function(err){
      console.log(err);
      $("#complete").hide();
      $("#fail").show();
      $("#transactionStatus").modal();
    });
  }
};
$(function() {
  $(window).load(function() {
    App.init();
  });
});
```

ชื่อไฟล์ : Test_FullData.sol

ภาษาที่ใช้ : Solidity

คำอธิบาย : smart contract ที่ใช้ในการจัดการฎีกาโฉนดที่ดินและการจำนอง

```
pragma solidity >=0.4.0 <0.6.0;

contract Test_FullData {

  uint private transactionNumber = 0;

  //โครงสร้างโฉนดที่ดิน(หน้าแรก)  ในทางปฏิบัติเลขที่โฉนดมันจะซ้ำตามอำเภอ ดังนั้นอาจจะมีกา
ระบุkeyเพิ่ม
  struct document {
    address id;
    //1. ตำแหน่งที่ดิน
    uint section; //ระวาง
    uint numberOfsection; //เลขที่ดิน
    string district; //ตำบล
    //2.โฉนดที่ดิน
    uint docNo; // เลขที่โฉนด
    string county; //อำเภอ
    string province; //จังหวัด
    //3.ระบุตัวเจ้าของ
    string name; // เอา frisname+lastname
    uint identificationNumber; //สัญชาติ
    //4.ที่ดินนี้มีเนื้อแหลงขนาด
    uint size;
    //5.รูปส่วนแผนที่>รูปแผนที่ มาตราส่วน ตำแหน่งทิศ วันที่ออก ลงชื่อเจ้าพนักงานที่ดิน
  }
  mapping(address => document[]) private allDocument;
```

```
//โครงสร้างโฉนดที่ดิน(หน้าหลัง)
struct transaction {
  uint traNo;
  //..other data
  uint docNo; // เลขที่โฉนด
  address mortgage; // ผู้ให้จำนอง
  address mortgagee; // ผู้รับจำนอง
  uint money; // จำนวนเงิน
  uint interest; // เงื่อนไขการจำนอง
  uint fee; //ค่าธรรมเนียม
}
mapping(uint => transaction[]) private allList;
transaction[] private waitingList;

//โครงสร้างโฉนดที่ดิน(หน้าหลัง)
struct endtransaction {
  uint traNo;
  //..other data
  uint docNo; // เลขที่โฉนด
  bool status;
}
mapping(uint => endtransaction) private allEndTransaction;

//ลงทะเบียนโฉนดที่ดินให้กับผู้ใช้
function sendDocument(address id, uint section, uint numberOfsection, string memory
district, uint docNo, string memory county, string memory province, string memory
name, uint identificationNumber, uint size) public {
  //address กรมที่ดิน
```

```
    require(msg.sender == 0x2B8F79184073919Db5b037eEf0a9Cf3f0B20910F);

    document memory newDocument = document(id, section, numberOfsection, district,
docNo, county, province, name, identificationNumber, size);
    allDocument[id].push(newDocument);
  }

  //เก็บธุรกรรมรอการอนุมัติ
  function saveTransaction(uint _docNo, address _mortgage, address _mortgagee, uint
_money, uint interest, uint fee) public {

    require(msg.sender == 0x723C6DB5E4A01B647Eb61e1094032eea969D9626);

    require(checkContract(_docNo));

    transactionNumber = transactionNumber + 1;
    transaction memory newTransaction = transaction(transactionNumber, _docNo,
_mortgage, _mortgagee, _money, interest, fee);
    waitingList.push(newTransaction);

  }

  //บันทึกธุรกรรมของโฉนดที่ดิน
  function approveTransaction(uint index, uint _docNo, address _mortgage, address
_mortgagee, uint _money, uint interest, uint fee) public {

    require(msg.sender == 0x2B8F79184073919Db5b037eEf0a9Cf3f0B20910F);
```

```
    if (index >= waitingList.length) return;

    transaction memory newTransaction = transaction(transactionNumber, _docNo,
_mortgage, _mortgagee, _money, interest, fee);
    allList[_docNo].push(newTransaction);

    for(uint i = index; i < waitingList.length-1; i++){
      waitingList[i] = waitingList[i+1];
    }
    delete waitingList[waitingList.length-1];
    waitingList.length--;
  }

  //ไม่อนุมัติสัญญา
  function rejectTransaction(uint index) public {
    require(msg.sender == 0x2B8F79184073919Db5b037eEf0a9Cf3f0B20910F);

    if (index >= waitingList.length) return;

    for(uint i = index; i < waitingList.length-1; i++){
      waitingList[i] = waitingList[i+1];
    }
    delete waitingList[waitingList.length-1];
    waitingList.length--;
  }

  //ยกเลิกพันธะ
  function endTransaction(uint _docNo, uint _traNo) public {
```

```
    //require
    require(msg.sender == 0x723C6DB5E4A01B647Eb61e1094032eea969D9626);
    //ตรวจสอบความถูกต้อง
    transaction[] memory tmpTran = allList[_docNo];

    bool status = false;
    for (uint idx = 0; idx < tmpTran.length; idx++){
      if(tmpTran[idx].traNo == _traNo && !allEndTransaction[tmpTran[idx].traNo].status){
        status = true;
      }
    }

    require(status);

    endtransaction memory newEndTransaction =  endtransaction(_docNo, _traNo, true);
    allEndTransaction[_traNo] = newEndTransaction;
  }
  //แสดงรายละเอียดของโฉนด
  function getDocument(uint _docNo) public view returns (uint, uint, string memory, uint,
string memory, string memory, string memory, uint, uint){
    //authen by user...
    document[] memory tmpDoc = allDocument[msg.sender];
    //require

    for(uint idx = 0; idx < tmpDoc.length; idx++){
      if(tmpDoc[idx].docNo == _docNo){

return(tmpDoc[idx].section,tmpDoc[idx].numberOfsection,tmpDoc[idx].district,tmpDoc[idx]
```

```
.docNo,tmpDoc[idx].county,tmpDoc[idx].province,tmpDoc[idx].name,tmpDoc[idx].identific
ationNumber,tmpDoc[idx].size);
      }
    }
  }
  //แสดงธุรกรรมของโฉนด
  function getTransaction(uint _docNo) public view returns (uint[] memory, address[]
memory, uint[] memory, bool[] memory, uint[] memory, address[] memory, uint[]
memory){
    //authen by user...
    address docOwner;
    for(uint idx = 0; idx < allDocument[msg.sender].length; idx++){
      if(allDocument[msg.sender][idx].docNo==_docNo){
        docOwner = allDocument[msg.sender][idx].id;
      }
    }
    //require
    //require(docOwner == msg.sender);

    transaction[] memory tmpTran = allList[_docNo];
    uint[] memory traNo = new uint[](tmpTran.length);
    address[] memory mortgagee = new address[](tmpTran.length);
    address[] memory mortgage = new address[](tmpTran.length);
    uint[] memory money = new uint[](tmpTran.length);
    bool[] memory status = new bool[](tmpTran.length);
    uint[] memory fee = new uint[](tmpTran.length);
    uint[] memory interest = new uint[](tmpTran.length);
    for(uint idx = 0; idx < tmpTran.length; idx++){
```

```
    traNo[idx] = tmpTran[idx].traNo;
    mortgagee[idx] = tmpTran[idx].mortgagee;
    mortgage[idx] = tmpTran[idx].mortgage;
    money[idx] = tmpTran[idx].money;
    fee[idx] = tmpTran[idx].fee;
    interest[idx] = tmpTran[idx].interest;
    if(allEndTransaction[tmpTran[idx].traNo].status){
      status[idx] = true;
     } else {
       status[idx] = false;
     }
    }
    return(traNo, mortgagee, money, status, fee, mortgage, interest);
  }

  //แสดงธุรกรรมของโฉนดที่รอการอนุมัติ
  function getWaitingList() public view returns (uint[] memory, address[] memory, uint[]
memory, uint[] memory, uint[] memory, address[] memory, uint[] memory){

    require(msg.sender == 0x2B8F79184073919Db5b037eEf0a9Cf3f0B20910F);

    uint[] memory traNo = new uint[](waitingList.length);
    uint[] memory docNo = new uint[](waitingList.length);
    address[] memory mortgagee = new address[](waitingList.length);
    address[] memory mortgage = new address[](waitingList.length);
    uint[] memory money = new uint[](waitingList.length);
    uint[] memory fee = new uint[](waitingList.length);
    uint[] memory interest = new uint[](waitingList.length);
```

```
  for(uint idx = 0; idx < waitingList.length; idx++){
    traNo[idx] = waitingList[idx].traNo;
    docNo[idx] = waitingList[idx].docNo;
    mortgagee[idx] = waitingList[idx].mortgagee;
    mortgage[idx] = waitingList[idx].mortgage;
    money[idx] = waitingList[idx].money;
    fee[idx] = waitingList[idx].fee;
    interest[idx] = waitingList[idx].interest;
  }
  return(traNo, mortgagee, money, docNo, fee, mortgage, interest);
}

//ตรวจสอบว่าไม่ติดพันธะ
function checkContract(uint _docNo) private view returns(bool) {
  //authen by user...
  //require
  transaction[] memory tmpTran = allList[_docNo];

    for(uint idx = 0; idx < waitingList.length; idx++){
      if(waitingList[idx].docNo == _docNo){
        return false;
      }
    }
    if(tmpTran.length == 0){
      return true;
    }
    if(!allEndTransaction[tmpTran[tmpTran.length - 1].traNo].status){
      return false;
```

```
        }
        return true;
    }
}
```

ชื่อไฟล์ : dolToken.sol

ภาษาที่ใช้ : Solidity

คำอธิบาย : smart contract ที่เป็น token ใช้ในการจัดการเสถียรของค่าธรรมเนียม

```
pragma solidity >=0.4.22 <0.6.0;

interface tokenRecipient {
    function receiveApproval(address _from, uint256 _value, address _token, bytes calldata
_extraData) external;
}

contract TokenERC20 {
    // Public variables of the token
    string public name;
    string public symbol;
```

```
    uint8 public decimals = 18;
    // 18 decimals is the strongly suggested default, avoid changing it
    uint256 public totalSupply;

    // This creates an array with all balances
    mapping (address => uint256) public balanceOf;
    mapping (address => mapping (address => uint256)) public allowance;

    // This generates a public event on the blockchain that will notify clients
    event Transfer(address indexed from, address indexed to, uint256 value);

    // This generates a public event on the blockchain that will notify clients
    event Approval(address indexed _owner, address indexed _spender, uint256 _value);

    // This notifies clients about the amount burnt
    event Burn(address indexed from, uint256 value);

    /**
     * Constructor function
     *
     * Initializes contract with initial supply tokens to the creator of the contract
     */
    constructor() public {
        totalSupply = 1000000 * 10 ** uint256(decimals);  // Update total supply with the decimal
amount
        balanceOf[msg.sender] = totalSupply;                // Give the creator all initial tokens
        name = "bolToken";                                  // Set the name for display purposes
```

```
    symbol = "DOL";                         // Set the symbol for display purposes
}

/**
 * Internal transfer, only can be called by this contract
 */
function _transfer(address _from, address _to, uint _value) internal {
    // Prevent transfer to 0x0 address. Use burn() instead
    require(_to != address(0x0));
    // Check if the sender has enough
    require(balanceOf[_from] >= _value);
    // Check for overflows
    require(balanceOf[_to] + _value >= balanceOf[_to]);
    // Save this for an assertion in the future
    uint previousBalances = balanceOf[_from] + balanceOf[_to];
    // Subtract from the sender
    balanceOf[_from] -= _value;
    // Add the same to the recipient
    balanceOf[_to] += _value;
    emit Transfer(_from, _to, _value);
    // Asserts are used to use static analysis to find bugs in your code. They should never fail
    assert(balanceOf[_from] + balanceOf[_to] == previousBalances);
}

/**
 * Transfer tokens
 *
```

```
     * Send `_value` tokens to `_to` from your account
     *
     * @param _to The address of the recipient
     * @param _value the amount to send
     */
    function transfer(address _to, uint256 _value) public returns (bool success) {
        _transfer(msg.sender, _to, _value);
        return true;
    }

    /**
     * Transfer tokens from other address
     *
     * Send `_value` tokens to `_to` on behalf of `_from`
     *
     * @param _from The address of the sender
     * @param _to The address of the recipient
     * @param _value the amount to send
     */
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool
success) {
        require(_value <= allowance[_from][msg.sender]);     // Check allowance
        allowance[_from][msg.sender] -= _value;
        _transfer(_from, _to, _value);
        return true;
    }
```

```
    /**
     * Set allowance for other address
     *
     * Allows `_spender` to spend no more than `_value` tokens on your behalf
     *
     * @param _spender The address authorized to spend
     * @param _value the max amount they can spend
     */
    function approve(address _spender, uint256 _value) public
        returns (bool success) {
        allowance[msg.sender][_spender] = _value;
        emit Approval(msg.sender, _spender, _value);
        return true;
    }

    /**
     * Set allowance for other address and notify
     *
     * Allows `_spender` to spend no more than `_value` tokens on your behalf, and then ping
the contract about it
     *
     * @param _spender The address authorized to spend
     * @param _value the max amount they can spend
     * @param _extraData some extra information to send to the approved contract
     */
    function approveAndCall(address _spender, uint256 _value, bytes memory _extraData)
        public
```

```
    returns (bool success) {
    tokenRecipient spender = tokenRecipient(_spender);
    if (approve(_spender, _value)) {
        spender.receiveApproval(msg.sender, _value, address(this), _extraData);
        return true;
    }
}

/**
 * Destroy tokens
 *
 * Remove `_value` tokens from the system irreversibly
 *
 * @param _value the amount of money to burn
 */
function burn(uint256 _value) public returns (bool success) {
    require(balanceOf[msg.sender] >= _value);   // Check if the sender has enough
    balanceOf[msg.sender] -= _value;            // Subtract from the sender
    totalSupply -= _value;                      // Updates totalSupply
    emit Burn(msg.sender, _value);
    return true;
}

/**
 * Destroy tokens from other account
 *
 * Remove `_value` tokens from the system irreversibly on behalf of `_from`.
```

```
     *
     * @param _from the address of the sender
     * @param _value the amount of money to burn
     */
    function burnFrom(address _from, uint256 _value) public returns (bool success) {
        require(balanceOf[_from] >= _value);                // Check if the targeted balance is enough
        require(_value <= allowance[_from][msg.sender]);    // Check allowance
        balanceOf[_from] -= _value;                         // Subtract from the targeted balance
        allowance[_from][msg.sender] -= _value;             // Subtract from the sender's allowance
        totalSupply -= _value;                              // Update totalSupply
        emit Burn(_from, _value);
        return true;
    }
}
```