การพัฒนาโมบายแอปพลิเคชันโดยใช้เทคโนโลยีบล็อกเชน

MOBILE APPLICATION USING BLOCKCHAIN TECHNOLOGY

ชญานิน จันทนานนท์

ณัฏฐนันท์ สายวรรณ์

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2562

การพัฒนาโมบายแอปพลิเคชันโดยใช้เทคโนโลยีบล็อกเชน

นายชญานิน จันทนานนท์ 59010255
นางสาวณัฏฐนันท์ สายวรรณ์ 59010404
ผศ.ดร. สมศักดิ์ วลัยรัชต์ อาจารย์ที่ปรึกษา
ผศ.ดร. อรัญญา วลัยรัชต์ อาจารย์ที่ปรึกษาร่วม
ปีการศึกษา 2562

บทคัดย่อ

ทุกวันนี้ทรัพย์สิน เช่น เงินสด โฉนดที่ดิน สัญญาซื้อขาย หรือเอกสารสำคัญอื่นๆมีการเก็บบันทึกในรูปแบบของตัวกลางที่สามารถจับต้องได้ แต่ทรัพย์สินเหล่านี้อาจจะสูญหาย เสียหาย หรือถูกปลอมแปลงโดยผู้ไม่ประสงค์ดีได้ จึงได้มีการคิดค้นวิธีที่จะดูแลรักษาทรัพย์สินให้ปลอดภัยหรือลดอัตราการเสียหายของทรัพย์สินให้มากที่สุด ปริญญานิพนธ์นี้จึงได้มีการจัดทำขึ้นเพื่อนำเสนอการประยุกต์ใช้เทคโนโลยีการบันทึกข้อมูลที่ขึ้นชื่อว่าปลอดภัยและโปร่งใสแก่การบันทึกข้อมูลที่สุด นั่นคือการบันทึกข้อมูลทรัพย์สินด้วยเทคโนโลยีบล็อกเชน โดยวิธีการนี้สามารถแก้ไขปัญหาข้างต้นได้ ด้วยการเขียนโค้ดบันทึกข้อมูลของกิจกรรมทางธุรกรรมใส่ลงในเครือข่ายบล็อกเชน และด้วยความสามารถพิเศษคือข้อมูลที่ถูกบันทึกไว้จะถูกบันทึกตลอดไป ไม่สามารถจะปลอมแปลงได้ ผู้จัดทำมองเห็นความสามารถของเทคโนโลยีนี้จึงได้พัฒนาแอปพลิเคชันมือถือในรูปแบบของเกมที่ประยุกต์ใช้เทคโนโลยีบล็อกเชนบนเครือข่ายอีเธอเรียมที่นิยมใช้กันโดยทั่วไป โดยแอปพลิเคชันจะเป็นรูปแบบของเกมสะสมสินทรัพย์ดิจิทัลซึ่งเรากำหนดให้แมวเป็นสินทรัพย์และสร้างมูลค่าของทรัพย์สินด้วยความหลากหลายทางพันธุกรรมของแมว รวมถึงสามารถเพิ่มมูลค่าของสินทรัพย์โดยการเพาะพันธุ์ ซื้อขาย แลกเปลี่ยนหรือการประมูล รวมถึงการให้เช่าสินทรัพย์ดิจิทัลนี้ได้

Mobile Application using Blockchain Technology

Mr. Chayanin	Chanthananond	59010255
Ms. Nattanan	Saiwan	59010404
Asst. Prof. Dr. Somsak	Walairacht	Advisor
Asst. Prof. Dr. Aranya	Walairacht	Co-Advisor

ABSTRACT

Normally, assets such as cash, title deeds, contracts, or other important documents are recorded in the form of tangible intermediaries. But these assets may be lost, damaged or forged by some people with malicious intents. Therefore, methods to maintain the property's safe or reduce the damage are invented. This project proposes an application of data recording technology that is safe and transparent to the data by using blockchain technology. The system can be used to solve the mentioned problems by writing code to record the transaction of data activities into the blockchain network with special abilities, that is the recorded data will be recorded forever and they cannot be changed. In this system, the capability of this technology is proposed in a form of a mobile game that use blockchain technology on the Etherium network. User can create a form of a digital asset collector game, which we have defined artificial cats as assets. Genetic diversities such as breeding, trading, exchanging, bidding and renting represent the attributes of digital assets.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จลุล่วงได้ด้วยความช่วยเหลือจากหลายฝ่ายทั้งในทางตรงและทางอ้อม ปริญญานิพนธ์ฉบับนี้จะสำเร็จลงไม่ได้หากปราศจากความช่วยเหลือของบุคคลเหล่านี้

ขอขอบคุณ อาจารย์ที่ปรึกษา ผศ.ดร.สมศักดิ์ วลัยรัชต์ และอาจารย์ที่ปรึกษาร่วม ผศ.ดร. อรัญญา วลัยรัชต์ ที่เป็นผู้ให้คำแนะนำและให้ความช่วยเหลือตลอดการทำปริญญานิพนธ์ฉบับนี้ซึ่งทำให้การทำงานต่างๆ เป็นไปได้อย่างราบรื่น

ขอขอบคุณอาจารย์และบุคลากรต่าง ๆ ในสาขาวิชาวิศวกรรมคอมพิวเตอร์ที่ได้ให้คำแนะนำมาโดยตลอด

สุดท้ายนี้ ขอขอบคุณ บิดา มารดา และครอบครัวที่ได้เลี้ยงดู สั่งสอน และให้การสนับสนุน พร้อมทั้งให้โอกาสในการศึกษาและให้กำลังในการทำงานเสมอมา

ชญานิน จันทนานนท์

ณัฏฐนันท์ สายวรรณ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูป **หน้า**

สารบัญรูป (ต่อ)

รูป **หน้า**

สารบัญรูป (ต่อ)

รูป **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาของปัญหา

ผู้จัดทำมองเห็นถึงปัญหาการเก็บสินทรัพย์หรือเอกสารสำคัญที่มีมูลค่าในปัจจุบัน ซึ่งเป็นสิ่งที่จับต้องได้จึงเกิดปัญหาการจัดเก็บและดูแลเป็นเหตุให้เกิดการสูญหาย เสียหาย หรือถูกปลอมแปลงโดยผู้ไม่ประสงค์ดี ในปัจจุบันซึ่งเป็นยุคของเทคโนโลยีดิจิทัล จึงมีเทคโนโลยีเกิดขึ้นมาใหม่ที่ชื่อว่า บล็อกเชน เป็นเทคโนโลยีที่ช่วยเก็บข้อมูลได้ปลอดภัยมากขึ้นและไม่จำเป็นต้องมีคนกลางในการดูแล แต่เนื่องจากว่าเทคโนโลยีนี้เป็นเทคโนโลยีใหม่และทำความเข้าใจได้ยาก ทางผู้จัดทำได้ทราบถึงปัญหาเหล่านี้ จึงต้องการที่จะพัฒนาแอปพลิเคชันมือถือในรูปแบบของเกมสะสมสินทรัพย์ประยุกต์กับเทคโนโลยีบล็อกเชนบนเครือข่ายอีเธอเรียม เพื่อที่จะสามารถทำให้บุคคลทั่วไปได้มีความเข้าใจและเข้าถึงเทคโนโลยีนี้ได้มากขึ้น

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1) เพื่อนำเสนอให้ผู้ที่มีความต้องการใช้เทคโนโลยีบล็อกเชนให้เข้าใจง่ายผ่านแอปพลิเคชันรูปแบบเกมบนโทรศัพท์มือถือ
2) เพื่อเป็นต้นแบบของการสร้างสินทรัพย์ดิจิทัลบนเทคโนโลยีบล็อกเชน
3) เพื่อศึกษาเกี่ยวกับการพัฒนาแอปพลิเคชันบนเทคโนโลยีบล็อกเชน
4) เพื่อศึกษาการพัฒนาแอปพลิเคชันบนระบบปฏิบัติการแอนดรอยด์
5) เพื่อเป็นแนวทางทางในการต่อยอดการพัฒนาแอปพลิเคชันบนเครือข่ายอีเธอเรียม

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1) แอปพลิเคชันนี้ทำให้ผู้จัดทำเข้าใจองค์ประกอบและการพัฒนาแอปพลิเคชันเทคโนโลยีบล็อกเชนบนเครือข่ายอีเธอเรียมมากขึ้น
2) บุคคลทั่วไปเข้าถึงเทคโนโลยีบล็อกเชนได้ง่ายขึ้น

3) ปริญญานิพนธ์นี้เป็นตัวอย่างสำหรับพัฒนาแอปพลิเคชันประเภทเกมบนโทรศัพท์มือถือโดยใช้เทคโนโลยีบล็อกเชน
4) ผู้ใช้งานได้รับความรู้ ความเข้าใจ เกี่ยวกับเทคโนโลยีบล็อกเชนมากขึ้น

1.4 วิธีดำเนินงาน

วิธีการดำเนินงานปริญญานิพนธ์นี้แบ่งออกเป็น 9 ขั้นตอนหลักดังนี้

1) กำหนดปัญหาที่จะทำในปริญญานิพนธ์
2) วิเคราะห์เพื่อหาแนวทางแก้ไขปัญหา
3) ศึกษาและรวบรวมทฤษฎีที่จะใช้ในการแก้ปัญหา
4) ออกแบบแอปพลิเคชัน ทั้งเชิงตรรกะและรูปร่างหน้าตา
5) ออกแบบโครงสร้างของระบบ
6) พัฒนาแอปพลิเคชัน
7) ทดสอบเพื่อหาข้อบกพร่อง
8) นำไปใช้งานจริงและวัดผล
9) จัดทำเอกสารและคู่มือการใช้งาน

1.5 ขอบเขตของปริญญานิพนธ์

1) แอปพลิเคชันสามารถใช้งานด้วยระบบปฏิบัติการแอนดรอยด์รุ่นที่ 6 ขึ้นไป
2) ขณะใช้งานแอปพลิเคชันจะต้องมีการเชื่อมต่ออินเตอร์เน็ต
3) แอปพลิเคชันนี้เป็นแอปพลิเคชันที่ใช้เทคโนโลยีบล็อกเชนเฉพาะบนเครือข่ายอีเธอเรียม
4) แอปพลิเคชันนี้อยู่บนเครือข่าย Ropsten (Test Network) และ Main Network เท่านั้น
5) ผู้ใช้ต้องเก็บรหัส (Seed Phrase) ที่ใช้สำหรับกู้คืน Wallet ไว้เอง ถ้าหากสูญหายจะไม่สามารถกู้คืนได้
6) ผู้ใช้ที่ได้รับสิทธิ์ความเป็นเจ้าของสินทรัพย์ดิจิทัลโดยสมบูรณ์ ตามที่ระบุใน NFT License

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 Blockchain

เทคโนโลยีบล็อกเชนถูกคิดค้นขึ้นมาครั้งแรกในปี 2008 โดย satoshi nakamoto ซึ่งเป็นนามแฝงของคนหรือกลุ่มคนหนึ่ง satoshi ได้กล่าวถึงเทคโนโลยีบล็อกเชน ใน White paper ของ Bitcoin ว่าเป็นระบบเงินอิเล็กทรอนิกส์ที่สามารถทำธุรกรรมได้โดยไม่ต้องผ่านสถาบันการเงินใด ๆ ซึ่งหมายความว่าเทคโนโลยีบล็อกเชนคือระบบที่ไร้ศูนย์กลาง(Decentralized) และ satashi ได้กล่าวว่าประโยชน์หลักของเทคโนโลยีบล็อกเชนคือการแก้ปัญหา Double-Spending ซึ่งต้องอาศัยระบบแบบมีศูนย์กลางคอยให้ความน่าเชื่อถือ เช่น ต้องมีระบบของธนาคารคอยตรวจธุรกรรมทางการเงินทุกครั้ง ซึ่งอาจเกิดความล่าช้าและเปลืองทรัพยากร อีกทั้งระบบนั้นต้องความน่าเชื่อถือมากพอ เพื่อให้มั่นใจได้ว่าเงินที่ใช้ทำธุรกรรมไปนั้นไม่เกิดการสูญหายระหว่างทาง

ความน่าเชื่อถือของเทคโนโลยีบล็อกเชนถูกสร้างขึ้นมาด้วยหลักการการที่ว่า ข้อมูลที่อยู่บนบล็อกเชนไม่สามารถเปลี่ยนแปลงได้ โดยในเครือข่ายของคอมพิวเตอร์ที่ทุกเครื่องมีสำเนาของข้อมูลชุดนี้อยู่เหมือนกันหมด เมื่อมีการทำธุรกรรม หรือ Transaction เกิดขึ้นมาใหม่ Transaction นี้จะถูกตรวจสอบและยืนยันผ่านเครือข่ายคอมพิวเตอร์นี้ ด้วย Consensus Algorithm . Consensus Algorithm นี้คือวิธีการที่สร้างความน่าเชื่อถือให้กับธุรกรรมนี้ เพื่อเป็นการยืนยันว่า Transaction นี้เชื่อถือได้ และสามารถป้องกันปัญหาการจ่ายที่ซ้ำซ้อนหรือ Double Spending Problem และเทคโนโลยีบล็อกเชนนี้มีหลายประเภทแบ่งตามการจำกัดการเข้าถึงเครือข่าย

2.1.1.1 Double Spending Problem

Satoshi Nakamoto ได้กล่าวไว้ใน White paper ของ Bitcoin ว่าประโยชน์หลักของเทคโนโลยีบล็อกเชนคือการเข้ามาแก้ปัญหา Double Spending ที่ต้องอาศัยตัวกลางที่น่าเชื่อถือมายืนยันธุรกรรมที่เกิดขึ้น และปัญหานี้มีพื้นฐานมาจากข้อมูลดิจิทัลที่สร้างขึ้นมาปัจจุบันสามารถถูกทำสำเนาออกมาได้ง่าย เราสามารถอัพโหลดรูปของเราลงโซเชียลมีเดีย ซึ่ง

หมายความว่าเราทำสำเนารูปไปเก็บไว้ที่เซิร์ฟเวอร์ของโซเชียลมีเดียนั้น และคนอื่นๆก็สามาเรียกใช้สำเนารูปของราได้นับไม่ถ้วน จะเกิดอะไรขึ้นถ้าสินทรัพย์ที่มีค่า เช่น เงิน ก็สามารถถูกทำสำเนาออกมานับไม่ถ้วนเหมือนกัน มันคงไม่ใช่เรื่องดีที่ คนหนึ่งคนโอนเงินให้ใครอีกคนโดยที่คนนั้นไม่สูญเสียเงินจำนวนนั้นไป ตัวอย่างเช่น A มีเงินอยู่ 100 บาท ต้องการโอนไปให้ B 100 บาทและขณะเดียวกันก็โอนไปให้ C อีก 100 บาท ปัญหานี้เกิดขึ้นจากการที่ระบบไม่มีการตรวจสอบที่พอ และเงินดิจิทัลนั้นไม่เหมือนกับเงินจริง ตรงที่สามารถทำสำเนาออกมาได้ ซึ่งบล็อกเชน จะมาแก้ปัญหาเหล่านี้

2.1.1.2 Consensus Algorithm

เนื่องจากเทคโนโลยีบล็อกเชนเก็บข้อมูลเป็นสำเนาบนเครื่องคอมพิวเตอร์ในเครือข่าย เครื่องคอมพิวเตอร์นี้ถูกเรียกว่าโหนด (Node) ซึ่งจะสื่อสารแลกเปลี่ยนข้อมูลกันตลอดเวลา สมมติว่ามีเครื่องใดเครื่องหนึ่งปลอมแปลงข้อมูลชุดหนึ่งขึ้นมาและบอกกับโหนดอื่น ๆ ว่าข้อมูลนี้เป็นจริง จึงต้องมีอัลกอริทึมที่ใช้ยืนยันความน่าเชื่อถือของข้อมูลชุดนี้ เรียกว่า Consensus Algorithm หนึ่งใน Consensus Algorithm ที่เป็นที่นิยมคือ Proof of Work เป็นอัลกอริทึมที่ Bitcoin ใช้ หลักการของ Proof of Work คือพลังการประมวลของ Node ในการแก้ปัญหาหนึ่งที่ยากในการหาคำตอบ แต่ตรวจคำตอบได้ง่าย เช่นการหาค่าหนึ่งที่เมื่อใช้ฟังก์ชัน hash เช่น SHA-256 แล้วขึ้นต้นด้วยเลข 0 จำนวน N บิท ซึ่งอัลกอริทึม สามารถตรวจได้โดยการใช้ฟังก์ชัน hash แค่ครั้งเดียว

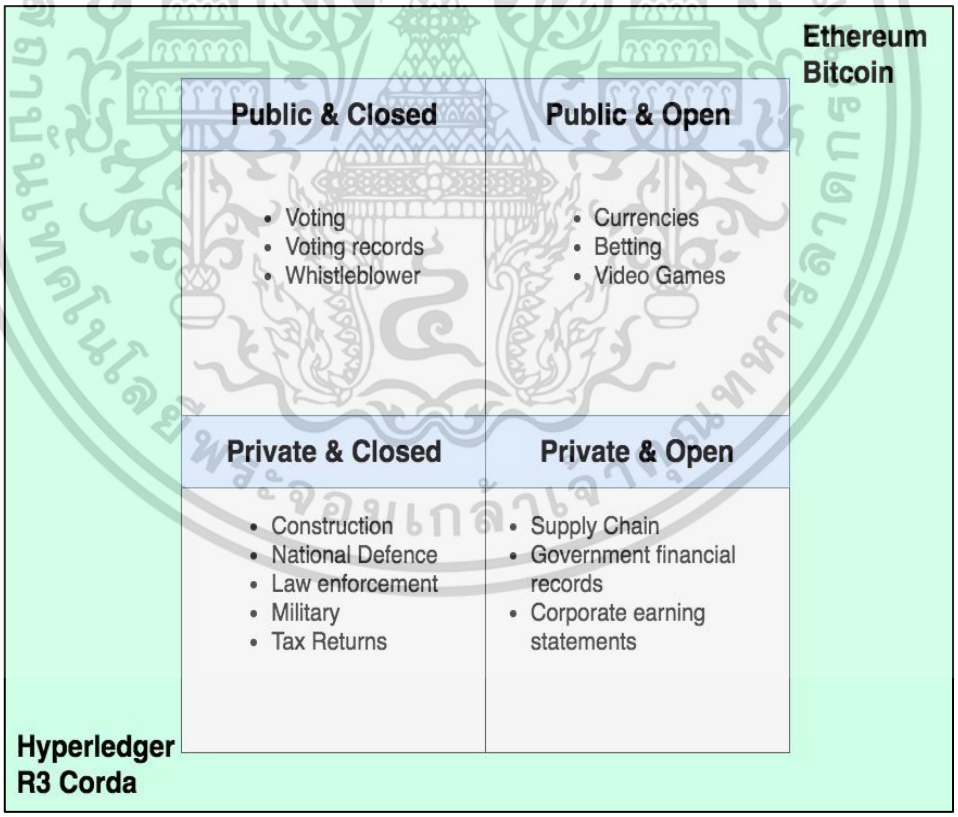
2.1.1.3 Transaction

วิธีการที่เราใช้ติดต่อกับบล็อกเชนอยู่ในรูปแบบของธุรกรรม (transaction) ผู้ที่เข้ามาใช้บล็อกเชนหรือสร้างธุรกรรมใหม่บนบล็อกเชนได้จำเป็นต้องมี address ซึ่งเปรียบได้กับบัญชีธนาคาร และคู่กุญแจ Public/Private Keys ในการยืนยันการทำธุรกรรมของบล็อกเชนนั้น ใช้หลักการของ Asymmetric Cryptography ซึ่งผู้สร้างธุรกรรมต้องยืนยันธุรกรรมด้วย private key ของตัวเอง ซึ่งเรียกว่า Digital Signatures ก่อนที่จะบันทึกลงเครือข่ายบล็อกเชน วิธีการนี้จะช่วยป้องกันการโจมตีจากผู้ไม่หวังดีได้ ซึ่งการจะทำธุรกรรมให้สำเร็จได้นั้นจำเป็นต้องมี private key ของเจ้าของธุรกรรมด้วย

2.1.1.4 ประเภทของบล็อกเชน

เราสามารถแบ่งประเภทของเทคโนลียีบล็อกเชนตามความสามารถในการอ่านและเขียนของผู้ใช้งานในเครือข่าย โดยถ้าแบ่งตามความสามารถในการเขียนออกได้เป็น 2 คือ Public Blockchain หรือ Permissionless ที่อนุญาตให้ใครก็ได้เขียนข้อมูลลงบนเครือข่ายและ Private Blockchain หรือ Permissioned ที่จำกัดให้แค่คนที่ได้รับอนุญาตเท่านั้นเขียนข้อมูลลงบนเครือข่ายได้ นอกจากนี้เราสามารถแบ่งตามความสามารถในการอ่านได้อีก 2 ประเภทคือ Open Blockchain และ Closed Blockchain สำหรับ Open Blockchain ใครก็ได้สามาอ่านข้อมูลได้ และ Closed Blockchain อนุญาตให้เฉพาะผู้ที่ได้รับอนุญาตอ่านเท่านั้น

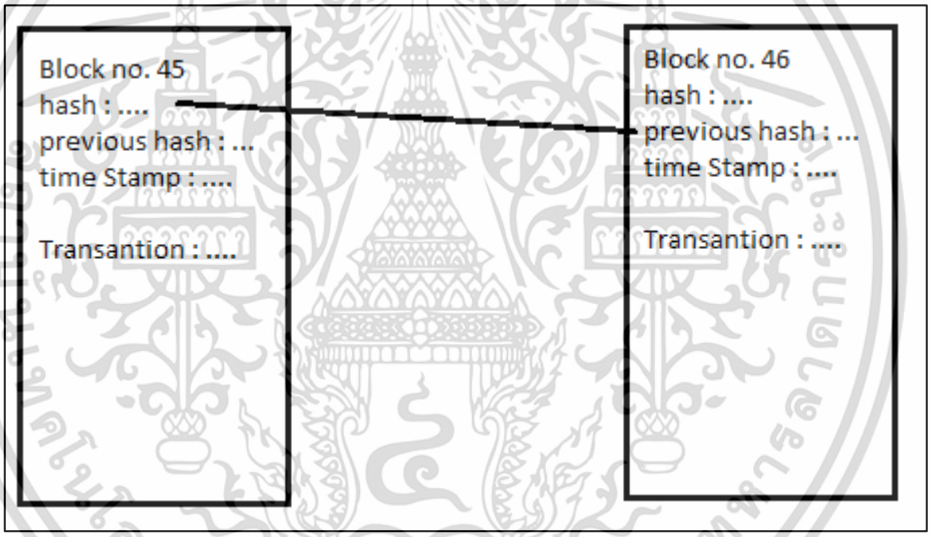
โดยทั่วไปคนรู้จักเทคโนโลยีบล็อกเชนในประเภทของ Open Public Blockchain ซึ่งเป็นรูปแบบของ Cryptocurrency อย่าง Bitcoin และ Ethereum ที่ 2.1 แสดงถึงการใช้งานบล็อกเชน



รูป 2.1 การใช้งานบล็อกเชน

2.1.1.5 การทำงานของบล็อกเชน

บล็อกเชน คือเครือข่าย peer-to-peer แบบกระจายศูนย์(Decentralized) ที่จะเก็บข้อมูลในรูปแบบกระจาย(Distributed) ซึ่งหมายความว่าจำเป็นต้องมีเครื่องคอมพิวเตอร์ในเครือข่ายที่มีสำเนาของข้อมูลนั้นอยู่มากกว่า 1 เครื่องและกลุ่มข้อมูลที่เก็บไว้นั้นเรียกว่า บล็อก (Blocks) แต่ละบล็อกที่เชื่อมต่อกันด้วยการเข้ารหัสทางเดียว (Cryptographic hash) ระบบจะทำการเข้ารหัสข้อมูลในบล็อกทั้งหมดด้วย hashing algorithm เช่น SHA256 แล้วจะได้ค่า hash ของบล็อกนั้นออกมา ซึ่งแต่ละบล็อกนั้น จะเก็บค่า hash ของก่อนหน้าไว้ด้วย กระบวนการนี้เรียกว่า Chaining โดยการ Chaining และการเก็บข้อมูลแบบกระจายนี้ ทำให้บล็อกเชนนั้นไม่สามารถเปลี่ยนแปลงได้ (Immutability)



รูป 2.2 โครงสร้างของ Block

เนื่องจากว่าบล็อกเชนเป็นระบบแบบกระจายศูนย์ ไม่เหมือนกับระบบแบบศูนย์กลางที่สามารถยืนยันการทำธุรกรรม(transaction)ได้ทันที เครือข่ายบล็อกเชนจึงจำเป็นต้องมีข้อตกลงในการยืนยันธุรกรรมหนึ่ง(Consensus protocol) ตัวอย่างเช่น Proof Of Work คือข้อตกลงหนึ่งที่ใช้ พลังของ CPU ในการยืนยันธุรกรรมนั้น กล่าวคือในแต่ละโหนด แข่งกันแก้ปัญหาปัญหาหนึ่งที่ยากต่อการแก้ แต่ง่ายต่อการตรวจสอบ เมื่อโหนดใดโหนดหนึ่งแก้ปัญหาได้สำเร็จก็จะมีสิทธิ์ยืนยันธุรกรรมนั้นลงบนบล็อกเชน และโหนดอื่นก็สามารถตรวจสอบได้ง่าย

2.1.1.6 Digital Assets

จากคุณสมบัติของบล็อกเชนที่ว่าข้อมูลภายในบล็อกไม่สามารถเปลี่ยนแปลงหรือทำซ้ำขึ้นมาได้ กล่าวคือไม่มีปัญหา Double Spending Problem เมื่อข้อมูลดิจิทัลไม่สามารถทำซ้ำขึ้นมาได้แล้วจึงทำให้เกิดสภาวะขาดแคลนของข้อมูลดิจิทัล (Digital Scarcity) ดังนั้นข้อมูลดิจิทัลเหล่านี้จึงมีมูลค่าขึ้นมา ตามหลักการของอุปสงค์อุปทาน ข้อมูลดิจิทัลเหล่านั้นถูกเรียกว่า สินทรัพย์ดิจิทัล (Digital Asset)

นิยามของสินทรัพย์บัญญัติศัพท์โดยสมาคมบัญชีและผู้สอบบัญชีรับอนุญาตแห่งประเทศไทยได้ให้ความหมายสินทรัพย์ว่า หมายถึง สิ่งที่มีตัวตนหรือหรือไม่มีตัวตน อันมีมูลค่าซึ่งบุคคลหรือกิจการเป็นเจ้าของ หรือมีสิทธิครอบครอง โดยถูกต้องตามกฎหมาย

สินทรัพย์ที่มีตัวตน (Tangible Assets) หมายถึง สินทรัพย์ที่สัมผัสได้ มองเห็นและมีตัวตน เช่น เงินสด เงินฝากธนาคาร ตั๋วเงินรับ ลูกหนี้ สินค้าคงเหลือ ที่ดิน อาหาร

สินทรัพย์ที่ไม่มีตัวตน (Intangible Assets) หมายถึง สินทรัพย์ที่ไม่อาจสัมผัสได้ไม่อาจแลเห็น แต่ให้ประโยชน์ต่อการดำเนินงานของกิจการ ซึ่งโดยส่วนใหญ่แล้วจะมีลักษณะเป็นสิทธิต่าง ๆ ตามกฎหมายที่กำหนด เช่น สัมปทาน สิทธิบัตร ลิขสิทธิ์ ค่าความนิยม เครื่องหมายการค้า สัญญาเช่า

สินทรัพย์ดิจิทัล (Digital Assets) คือสินทรัพย์ในรูปแบบของข้อมูลดิจิทัล โดยสำนักงานคณะกรรมการกำกับหลักทรัพย์และตลาดหลักทรัพย์ (ก.ล.ต.) ได้ให้นิยามของสินทรัพย์ดิจิทัลไว้ 2 ประเภท ตามพระราชกำหนดการประกอบธุรกิจสินทรัพย์ดิจิทัล คือ

1.) คริปโทเคอร์เรนซี (Cryptocurrency)

หมายความว่า เป็นหน่วยข้อมูลอิเล็กทรอนิกส์ซึ่งถูสร้างขึ้นบนระบบหรือเครือข่ายอิเล็กทรอนิกส์โดยมีความประสงค์ที่จะใช้เป็นสื่อกลางในการแลกเปลี่ยนเพื่อให้ได้มาซึ่งสินค้าบริการ หรือสิทธิอื่นใด หรือแลกเปลี่ยนระหว่างสินทรัพย์ดิจิทัล และให้หมายความรวมถึงหน่วยข้อมูลอิเล็กทรอนิกส์อื่นใดตามที่คณะกรรมการ ก.ล.ต. ประกาศกำหนด โดยคริปโคเคอร์เรนซีหรือสกุลเงินดิจิทัล เป็น สกุลเงินอิเล็กทรอนิกส์แบบ p2p สามารถโอนจากผู้ถือคนหนึ่งไปยังอีกคนได้โดยไม่ต้องผ่านสถาบันการเงินใดๆ และคริปโคเคอร์เรนซีนี้สร้างอยู่

บนเทคโนโลยีบล็อกเชน ซึ่งไม่สามารถถูกควบคุมโดยคนหรือกลุ่มคนเพียงหนึ่งเดียว ตัวอย่างของคริปโทเคอร์เรนซีที่มีอยู่ในปัจุบัน เช่น บิทคอยน์ (BTC) , อีเธอ (ETH) , ริบเปิล (XRP) , บิทคอยน์ แคช (BCH) , ทีเธอ (USDT)

2.) โทเคนดิจิทัล (Initial Coin Offering)

หมายความว่า หน่วยข้อมูลอิเล็กทรอนิกส์ซึ่งถูกสร้างขึ้นบนระบบหรือเครือข่ายอิเล็กทรอนิกส์โดยมีวัตถุประสงค์เพื่อ กำหนดสิทธิของบุคคลในการเข้าร่วมโครงการหรือกิจการใดๆ และกำหนดสิทธิในการได้มาซึ่งสินค้าหรือบริการหรือสิทธิอื่นใดที่เฉพาะเจาะจง ทั้งนี้ตามกำหนดในข้อตกลงระหว่างผู้ออกและผู้ถือ และให้หมายความรวมถึงหน่วยแสดงสิทธิอื่นตามที่คณะกรรมการ ก.ล.ต. ประกาศกำหนด

2.1.2 Ethereum

เครือข่ายบล็อกเชนของ Bitcoin นั้นทำได้แค่เพียงเป็นสกุลเงินที่ใช่สำหรับใช้จ่ายเท่านั้น ไม่สามารถใช้เป็นเครือข่ายเพื่อให้ผู้พัฒนาสร้างแอปพลิเคชันได้ แต่ในปี ค.ศ. 2015 Ethereum ได้เปิดตัวขึ้นมาเพื่อเป็นเครือข่ายบล็อกเชน สำหรับให้นักพัฒนาแอปพลิเคชันลงบนเครือข่ายนี้ได้ Ethereum ให้ความสามารถในการประมวลผลและภาษาโปรแกรมขึ้นมาเอง

อีเธอเรียม (Ethereum) คือเครือข่ายบล็อกเชนสาธารณะ ที่อนุญาตให้คนทั่วไปใช้สำหรับสร้างแอปพลิเคชันบนเครือข่ายได้ โดยมีสกุลเงินดิจิทัลที่ชื่อว่า อีเธอ (ETH , Ether) เครือข่ายอีเธอเรียมนี้เป็นที่นิยมเป็นอย่างมากในการสร้างแอปพลิเคชันแบบกระจายศูนย์ (DApp , Decentralized Application) เนื่องจากเครือข่ายนี้เปิดเป็นสาธารณะและอนุญาตให้คนทั่วไปพัฒนาแอปพลิเคชันลงบนเครือข่ายอีเธอเรียม ด้วยการเขียน smart contract แล้ว Deploy ลงบนเครือข่าย ทำให้เครือข่ายอีเธอเรียมนั้นเป็นเครือข่ายแรกที่ใช้หลักการเขียน smart contract บนเครือข่ายบล็อกเชน

2.1.2.1 Ethereum Account

การที่จะติดต่อกับเครือข่ายอีเธอเรียมได้นั้นจำเป็นต้องมี account ซึ่งแบ่งออกเป็น 2 ประเภทคือ Externally Owned Accounts (EOAs) และ Smart Contract . EOAs หรือบัญชีผู้ใช้ภายนอก จะถูกควบคุมโดยผู้ใช้ผ่านทางแอปพลิเคชันที่ให้บริการ wallet จากภายนอกซึ่งผู้ให้บริการ Wallet นี้มีหน้าที่เก็บคู่กุญแจ Public/Private key ของผู้ใช้ไว้และให้บริการเซ็น

ธุรกรรมให้กับผู้ใช้งาน เครือข่ายอีเธอเรียมจะมอง address ทั้ง 2 ประเภทเป็น address หนึ่งที่เหมือนกัน

2.1.2.2 Smart Contract

สำหรับ Traditional Contract หรือสัญญาทั่วไปที่อยู่บนกระดาษ ใช้เพื่อสร้างข้อตกลงระหว่างบุคคลทั้งสองฝ่ายในทางกฎหมาย ซึ่งการเก็บรักษาสัญญารูปแบบนี้เก็บได้ยาก มีค่าดำเนินการสูงและล่าช้าเนื่องจากต้องใช้นักกฎหมาย โดยที่ Smart Contract ถูกสร้างขึ้นมาแทนที่สัญญาทั่วไปนี้ และแก้ปัญหาเหล่านี้

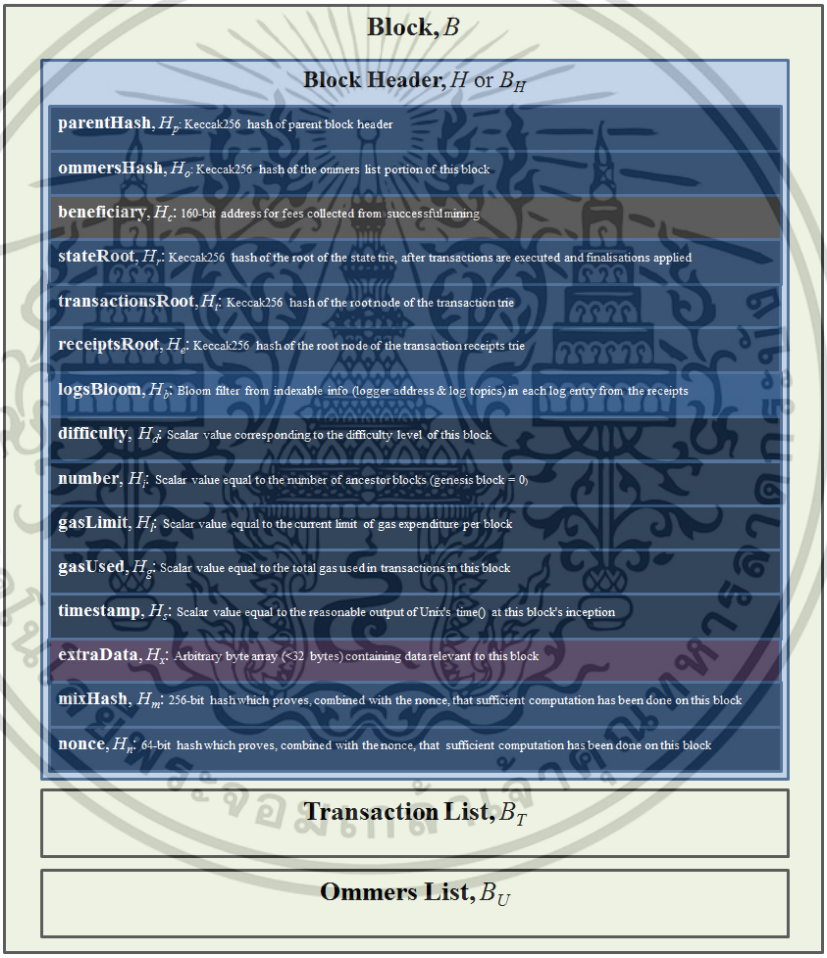
Smart Contract คือโค้ดของโปรแกรมที่จะทำงานบนเครือข่าย สามารถเขียนโดยใช้ภาษา Solidity และภาษาอื่นๆที่รองรับ โดย smart contract ที่ถูกเขียนด้วยภาษาเหล่านี้จะต้องถูกคอมไพล์และแปลงเป็น Ops code และถูกนำไปประมวลผล โดย Ethereum Virtual Machine (EVM) เมื่อนำไป Deploy บนเครือข่ายแล้วจะเรียกโค้ดนี้ว่า contract ซึ่งเทียบได้กับ class ในภาษาโปรแกรมเชิงวัตถุ . ใน contract จะมีทั้งข้อมูลสถานะ (state variable) และฟังก์ชัน ซึ่งการที่จะเปลี่ยนแปลงข้อมูลสถานะของ contract ก็หมายถึงการเปลี่ยนสถานะของบล็อกเชนด้วย

2.1.2.3 Node

เครือข่ายบล็อกเชน เป็นเครือข่ายแบบกระจายศูนย์ (Distributed Network) แต่ละเครื่องคอมพิวเตอร์ในเครือข่ายเรียกว่า โหนด ในแต่ละโหนดจะมีชุดสำเนาข้อมูลเดียวกันทั้งหมด และแต่ละโหนดติดต่อกันด้วยโปรโตคอล json-RPC (Remote Procedure Call) และเป็นโปรโตคอลเดียวกันกับที่ใช้ติดต่อกับ ฝั่ง Client เพื่อรับ Transaction

โหนด (Node) เป็นเหมือนตัวแทนของเครือข่ายอีเธอเรียม แต่ละโหนดในเครือข่ายนั้นจะมีสำเนาของข้อมูลเหมือนกันทั้งหมดทุกโหนด เป็นไปตามหลักการของบล็อกเชน และการที่เราจะสามารถสร้างโหนดขึ้นมาและเข้าร่วมเครือข่ายได้นั้น เราจำเป็นต้องใช้ client platform ที่ใช้สร้างและรันโหนดเข้าไปในเครือข่ายอีเธอเรียม และปัจจุบัน client เหล่านั้นมีให้เลือกหลากหลาย แบ่งตามภาษาที่ใช้เขียน เช่น Geth (Golang) , Parity (Rust) , Py-EVM (python) . โหนดนั้นมีอยู่ 3 ประเภทแบ่งตามขนาดของข้อมูลที่เก็บ 1. Full nodes เก็บข้อมูลทั้งหมดที่อยู่บนเครือข่าย (ซึ่งปัจจุบันใช้พื้นที่ประมาณ 220 GB) , รับธุรกรรมใหม่มาและส่วน

ร่วมในการ validate ธุรกรรมนั้น 2. Light nodes เก็บข้อมูลแค่ส่วนของ block header (รูปที่ 2.3) แต่ light nodes นั้นไม่ได้เชื่อมต่อกับเครือข่ายโดยตรง แต่มันเชื่อมต่อผ่าน Full nodes อีกที . การที่ light nodes สามารถเชื่อ Full nodes ที่เชื่อมอยู่ได้ เพราะ block header เก็บข้อมูลที่เรียกว่า merkle tree ซึ่งเป็นเหมือนลายนิ้วมือของข้อมูลทั้งหมดในบล็อกเชน ถ้าข้อมูลในบล็อกเชนเปลี่ยน merkle tree ก็จะเปลี่ยนด้วย ถ้าขอข้อมูลมาจาก Full nodes ก็จะสามารถตรวจสอบข้อมูลนั้นได้ว่าถูกต้องหรือไม่ 3. Archive nodes เหมือนกับ Full nodes แต่ต่างกันตรงที่ Archive nodes จะเก็บทุก state เอาไว้ทั้งหมด



รูป 2.3 Ethereum Block Header

2.1.2.4 Ethereum Virtual Machine

ในเครือข่ายอีเธอเรียม ไม่เพียงแต่สามารถบันทึกข้อมูลธุรกรรมได้เท่านั้น แต่ยังสามารถแชร์ความสามารถในการประมวลผลกันได้อีกด้วย ผ่านทาง smart contract ที่สามารถคอมไพล์และรันบน Node ด้วย Ethereum Virtual Machine . Ethereum Virtual Machine (EVM) คือ Virtual Machine สร้างชั้นที่ใช้ประมวลผล smart contract และตัวอีเธอเรียมบล็อกเชนเอง สำหรับภาษาหลักที่ใช้เขียน smart contract คือ Solidity และอีกหลายภาษา ซึ่งเป็นภาษาชั้นสูง EVM ไม่สามารถประมวลจากภาษานี้ได้โดยตรง จำเป็นต้องใช้ complier แปลงโค้ด solidity ไปเป็น Ethereum Opcodes ที่ EVM สามารถประมวลได้ . เนื่องจากการประมวลผลของทุก contract จะถูกประมวลผลโดยทุกโหนดที่อยู่บนเครือข่าย ผู้ที่ไม่หวังดีจึงสามารถโจมตีด้วยการสร้าง contract ที่ใช้การประมวลผลสูง เพื่อป้องกันการโจมตีนั้น ในแต่ละ opcodes จะมีค่าธรรมเนียมที่ใช้ในการประมวลผล(หรือที่เรียกว่า gas) ค่าธรรมเนียมนี้ผู้ที่เรียกใช้ smart contract ต้องเป็นผู้จ่ายสำหรับการประมวลนั้น

2.1.2.5 Ethereum Transaction

ธุรกรรม (Transaction) ถูกใช้ในการติดต่อกับเครือข่ายอีเธอเรียม เพื่อส่งอีเธอจาก Account ไป Account หรือใช้เรียกฟังก์ชันใน Smart Contract ให้ทำงาน หรือใช้สร้าง Smart Contract ขึ้นมาในเครือข่ายอีเธอเรียมด้วย โดยโครงสร้างของธุรกรรมในอีเธอเรียม ประกอบไปด้วย

1) From คือ ผู้ส่ง Transaction นั้นเป็น address ขนาด 20 byte

2) To คือ address ของผู้รับ อาจเป็น EOAs เมื่อใช้ธุรกรรมนี้สำหรับส่งอีเธอ หรือเมื่อเป็น address ของ Smart Contract ก็ใช้เพื่อเรียกฟังก์ชัน หรือเมื่อไม่มีอยู่ในส่วนนี้ ก็เป็นการสร้าง Smart Contract ขึ้นบนเครือข่าย

3) Data คือ ข้อมูลที่ใช้สำหรับเรียกใช้ฟังก์ชันใน Smart Contract หรือสร้าง Smart Contract

4) Gas Price คือ ค่าธรรมเนียมที่ต้องจ่ายในการยืนยันธุรกรรมนี้ ให้กับ Miner

Unit	Wei Value	Wei
wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
microether (szabo)	1e12 wei	1,000,000,000,000
milliether (finney)	1e15 wei	1,000,000,000,000,000
ether	1e18 wei	1,000,000,000,000,000,000

รูป 2.4 Ethereum gas table

2.1.2.6 Ethereum Consensus

ในเครือข่ายอีเธอเรียมเวอร์ชั่น 1.0 ใช้ Proof of Work เป็น consensus หมายความว่าจำเป็นต้องโหนดที่ทำหน้าที่คำนวณปัญหาที่ใช้ตรวจสอบและยืนยันธุรกรรมที่เข้ามาใหม่ เรียกโหนดนั้นว่า Miner และ miner ที่คำนวณปัญหาสำเร็จคนแรกจะรับรางวัลเป็น ETH เราเรียกกระบวนการนี้ว่า mining ซึ่งคำว่า mining นี้เลียนแบบมาจากการขุดทอง ซึ่งทองนั้นมีค่าเพราะว่ามันมีน้อยและจำกัด และมีทางเดียวที่จะเพิ่มปริมาณทองที่มีอยู่ได้คือการขุดหามัน (mining)

Mining Algorithm ที่ใช้กันในปัจจุบันเรียกว่า Ethash ซึ่งพัฒนามาจากอัลกอริทึมเก่าชื่อว่า Dagger Hashimoto อัลกอริทึมนี้ต้องการหาเลขสุ่มค่าหนึ่ง (Nonce) ใส่เข้าไปน Hashing Algorithm ที่ใช้ตรวจสอบความถูกต้องของบล็อก และ Nonce นั้นต้องให้ผลลัพธ์ที่ตรวจสอบง่ายที่สุด ตามหลักการของ Proof of Work ที่บอกว่า Mining Algorithm ต้องมีความยากในในการหา และง่ายต่อการตรวจสอบ เพราะฉะนั้น Mining Nodes จึงต้องต้องแข่งกันหาค่า Nonce ที่ทำให้ความยากในการตรวจสอบน้อยที่สุด และ Node แรก ที่ทำให้ค่าความยากน้อยกว่าค่าขีดจำกัด (Threshold) ที่เครือข่ายตั้งไว้ได้คนแรกจะได้รับรางวัล (Reward) ซึ่งเป็นการสร้าง ETH ขึ้นมาใหม่บนเครือข่าย จึงทำให้จำนวน ETH ทั้งหมดมากขึ้นเหมือนกับการขุดหาทอง

2.1.2.7 Ethereum Improvement Proposals (EIPs)

เนื่องจากอีเธอเรียมเป็นโปรเจค Open Source จึงจำเป็นต้องมีมาตรฐานสำหรับการ Contribute ให้กับโปรเจค ซึ่ง EIPs คือมาตรฐานสำหรับเครือข่ายอีเธอเรียม รวมไปถึง Core Protocol Specifications, Client APIs และมาตรฐานในการสร้าง Smart Contract

มาตรฐาน ERC-20 คือ มาตรฐานของ Smart Contract บนเครือข่ายเครือข่ายอีเธอเรียม สำหรับใช้สร้างคริปโตเคอร์เรนซี ปัจจุบันมีเหรียญ ERC-20 บนเครือข่ายอีเธอเรียมอยู่ประมาณ 217,759 เหรียญ มาตรฐาน ERC-20 กำหนดตัวแปรและฟังก์ชันไว้ดังรูปที่ 2.5

```
interface ERC20 {
//standard variables
  function name() public view returns ();

  function symbol() public view returns ();

  function decimals() public view returns (uint8);

  function totalSupply() public view returns (uint256);

//core ERC20 functions
  function balanceOf(address _owner) public view returns (uint256 balance);

  function transfer(address _to, uint256 _value) public returns (bool success);

  function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);

  function approve(address _spender, uint256 _value) public returns (bool success);

  function allowance(address _owner, address _spender) public view returns (uint256 remaining);

  event Transfer(address indexed _from, address indexed _to, uint256 _value);

  event Approval(address indexed _owner, address indexed _spender, uint256 _value);

}
```

รูป 2.5 ตัวแปรและฟังก์ชันพื้นฐานของ ERC-20

มาตรฐาน ERC-721 คือ มาตรฐานของ Smart Contract บนเครือข่ายเครือข่าย อีเธอเรียมสำหรับใช้สร้าง Non-Fungible Token กำหนดตัวแปรและฟังก์ชันไว้ดังรูปที่ 2.6

```
interface ERC721 {
    event Transfer(address indexed _from, address indexed _to, uint256 indexed _tokenId);
    event Approval(address indexed _owner, address indexed _approved, uint256 indexed _tokenId);
    event ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved);
    function balanceOf(address _owner) external view returns (uint256);
    function ownerOf(uint256 _tokenId) external view returns (address);
    function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data) external payable;
    function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable;
    function transferFrom(address _from, address _to, uint256 _tokenId) external payable;
    function approve(address _approved, uint256 _tokenId) external payable;
    function setApprovalForAll(address _operator, bool _approved) external;
    function getApproved(uint256 _tokenId) external view returns (address);
    function isApprovedForAll(address _owner, address _operator) external view returns (bool);
}
```

รูป 2.6 ตัวแปรและฟังก์ชันพื้นฐานของ ERC-721

2.1.3 ภาษาที่ใช้พัฒนา Smart Contract

EVM ของเครือข่ายอีเธอเรียมทำงานด้วยภาษาระดับล่าง เรียกว่า EVM Opcodes ในการทำงาน แต่มีภาษาระดับสูงที่สามารถถูกคอมไพล์ไปเป็น Opcodes เพื่อเขียน Smart Contract บนอีเธอเรียมได้ เช่นภาษา Solidity และ Vyper แต่ปัจจุบันภาษา Solidity ได้รับความนิยมมากที่สุด

Smart Contract ที่รันบนโหนดอีเธอเรียมถูกเขียนด้วย ภาษา Solidity ซึ่งถูกสร้างมาเพื่อคอมไพล์และรันบน EVM เป็นภาษาโปรแกรมเชิงวัตถุ (Object-oriented programming) มีพื้นฐานมีจากภาษา C++ , Python , Javascript กำหนดประเภทตัวแปลแบบสถิต (statically typed) รองรับ inheritance เราสามารถใช้ Solidity เขียน Smart Contract บนเครือข่ายอีเธอเรียม

2.1.3.1 Smart Contract อย่างง่าย

เริ่มต้นที่ smart contract อย่างง่าย เพื่อทำความเข้าใจโครงสร้างและการทำงาน

```
pragma solidity >=0.4.0 <0.7.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) public {
        storedData = x;
    }

    function get() public view returns (uint) {
        return storedData;
    }
}
```

รูป 2.7 Simple Storage Contract

บรรทัดแรกของโค้ดใช้เพื่อบอกว่า โค้ดนี้ถูกเขียนด้วย Solidity เวอร์ชั่น 0.4.0 หรือใหม่กว่า แต่ไม่รองรับเวอร์ชั่น 0.7.0 ขึ้นไป เพื่อให้แน่ว่าโค้ดนี้จะไม่ถูกคอมไพล์ด้วยคอมไพเลอร์ผิดเวอร์ชั่น ส่งผลให้เกิดปัญหาขึ้นภายหลังได้

Contract ในความหมายของ Solidity คือ ภาชนะที่เก็บโค้ด(Function) และข้อมูลสถานะ(State) ของ address หนึ่งในเครือข่ายอีเธอเรียม . บรรทัด **uint storedData;** ประกาศตัวแปรที่ชื่อว่า storedData ประเภท uint (unsigned integer 256 bit) ให้คิดว่าตัวแปรตัวนี้เหมือนช่องๆหนึ่งในฐานข้อมูลที่เราสามารถเรียกดูและเปลี่ยนแปลงมันได้ โดยการเรียกฟังก์ชันที่ควบคุมฐานข้อมูลนั้น . ดังเช่นในตัวอย่างนี้ ฟังก์ชันนั้นก็คือ **get** และ **set** ที่สามารถใช้เรียกดูและเปลี่ยนแปลงค่าของตัวแปลนั้นได้

ใน Contract ตัวอย่างนี้อนุญาตให้ใครก็ได้สามารถเข้าถึง Contract นี้ได้ ใครๆก็สามารถเรียกฟังก์ชัน get เพื่อเปลี่ยนแปลงค่า storedData ได้ แต่ประวัติการเปลี่ยนก็จะถูกบันทึกไว้ด้วย ตามการทำงานของบล็อกเชน

2.1.3.2 Subcurrency เงินดิจิทัลอย่างง่าย

ตัวอย่างที่นำมาประยุกต์ใช้และทำให้คนทั่วไปเห็นภาพมากที่สุดสำหรับเทคโนโลยีบล็อกเชน คือ สกุลเงินดิจิทัล ต่อไปนี้จะกล่าวถึงตัวอย่างการสร้างสุกลเงินดิจิทัลอย่างง่ายด้วยภาษา Solidity

รูปที่ 2.8 แสดง Contract Coin หรือสกุลเงินดิจิทัลอย่างง่าย ที่อนุญาตให้แค่คนที่สร้าง Contract นี้สร้างเหรียญใหม่ขึ้นหรือเงินเข้าไปในระบบเพิ่มได้ และใครๆก็สามารถส่งเหรียญ

นี้หากันได้โดยไม่ต้องทะเบียนเพิ่มเลย นอกจากมี address ของเครือข่ายอีเธอเรียมและ private ที่ใช้เซ็น transaction

```
pragma solidity >=0.5.0 <0.7.0;

contract Coin {

    address public minter;
    mapping (address => uint) public balances;

    event Sent(address from, address to, uint amount);

    constructor() public {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        require(amount < 1e60);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance.");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

รูป 2.8 Contract ของสกุลเงินดิจิทัลอย่างง่าย

บรรทัดที่มี **address public minter;** หมายถึง ประกาศตัวแปรข้อมูลสถานะของ Contract มีประเภทเป็น address ที่มีขนาด 160 bit และไม่สามารถใช้กับการดำเนินการทางพีชคณิต(arithmetic operations)ได้ ซึ่งตัวแปลประเภทนี้จะใช้เก็บ Address ของ Contract หรือเก็บ address ของผู้ใช้ภายนอก และคำว่า public หมายถึง Contract นี้อนุญาตให้เข้าถึงตัวแปรนี้จากภายนอก contract ได้โค้ดข้างต้นนี้เมื่อคอมไพล์แล้วจะมีการทำงานเหมือนฟังก์ชันในรูปที่ 2.9

```
function minter() external view returns (address) { return minter; }
```

รูป 2.9 ตัวอย่างฟังก์ชันอ่านค่าข้อมูลภายใน contract

ฟังก์ชันข้างบนทำงานได้เหมือนกับการเรียกดูค่าของตัวแปล minter แต่ก็ไม่จำเป็นเพราะว่าคอมไพเลอร์จะทำให้เราเอง บรรทัดต่อมาคือ **mapping (address => unint) public**

balances; ซึ่งเป็นตัวแปรข้อมูลสถานะเช่นเดียวกับ **minter** แต่มีโครงสร้างที่ซับซ้อนกว่า กล่าวคือตัวแปรประเภท mapping นี้สามารถเรียกว่าเหมือนกับ hash table . map ค่า address ด้วยค่า uint ค่าหนึ่ง ซึ่งตัวแปลประเภทนี้เป็นที่นิยมมากกว่าการใช้อาเรย์ทั่วไป

ฟังก์ชัน getter จะดูซับซ้อนขึ้นเมื่อใช้กับตัวแปร mapping ดังรูปที่ 2.10 และสามารถเรียกใช้มันเพื่อดูค่า balances ของ account ใน mapping ได้

```
function balances(address _account) external view returns (uint) {
    return balances[_account];
}
```

รูป 2.10 ฟังก์ชัน getter ที่ใช้กับตัวแปรประเภท mapping

บรรทัดที่บอกว่า **event Send(address from,address to, uint amount);** ใช้ประกาศ event ซึ่งถูกใช้(emitted) ในบรรทัดสุดท้าย ของฟังก์ชัน send . Ethereum client เช่นเว็บแอป-พลิเคชัน สามารถสังเกต (listen) event นี้ได้โดยที่ไม่ต้องเสียทรัพยากรมาก ถ้า event นี้เกิดขึ้นมา , listener จะได้รับ argument ของ event นั้นมา ซึ่งมี from , to , amount ทำให้เราสามารถติดตามธุรกรรมนี้ได้

เพื่อที่จะ listen event นี้ เราสามารถใช้ภาษา javascript โดยใช้ library ที่ชื่อว่า web3.js สร้าง contract object ขึ้นมาเพื่อใช้ติดต่อกับ contract นี้ในเครือข่ายอีเธอเรียม ดังรูปที่ 2.11

```
Coin.Sent().watch({}, '', function(error, result) {
    if (!error) {
        console.log("Coin transfer: " + result.args.amount +
            " coins were sent from " + result.args.from +
            " to " + result.args.to + ".");
        console.log("Balances now:\n" +
            "Sender: " + Coin.balances.call(result.args.from) +
            "Receiver: " + Coin.balances.call(result.args.to));
    }
})
```

รูป 2.11 โปรแกรมภาษา javascript ที่ใช้ listen event Send

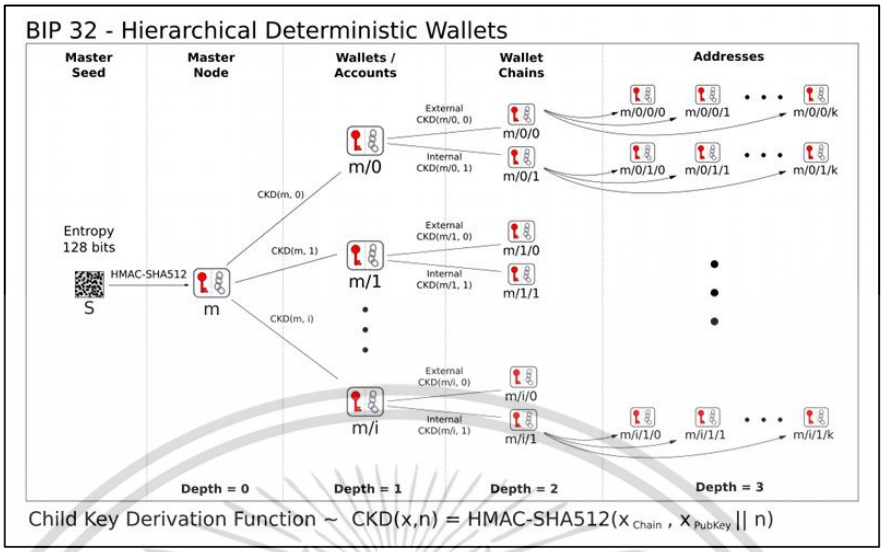
Constructor คือฟังก์ชันพิเศษที่ใช้สร้าง contract นี้ขึ้นมา ซึ่งถูกเรียกใช้ขณะสร้าง แค่ครั้งเดียว สำหรับ contract นี้มันจะบันทึก address ของผู้ที่สร้าง contract นี้ลงตัวแปล minter .

ดังบรรทัดที่มี **minter = msg.sender;** ซึ่งตัวแปล msg เป็นตัวแปลโกลบอลพิเศษที่สามารถทำให้เราเข้าถึงเครือข่ายได้ และ msg.sender จึงหมายถึง address ของผู้ใช้ภายนอกที่เรียกใช้ contract นี้ ซึ่งในที่นี้หมายถึงคนที่สร้าง contract นี้ขึ้นมา และใน contract มีอีก 2 ฟังก์ชันที่ยังไม่กล่าวถึง คือ mint และ send 1. mint คือการเพิ่มจำนวนเหรียญใหม่ขึ้นมาใน contract ไปให้ address หนึ่งซึ่งมีเงื่อนไขอยู่ว่าผู้ที่จะเรียกฟังก์ชันนี้ได้ต้องเป็น minter เท่านั้น require(msg.sender == minter) และไม่สามารถสร้างเพิ่มขึ้นได้มากกว่าจำนวนที่กำหนดไว้ 2. send สามารถถูกเรียกใช้โดยใครก็ได้ (ที่มีเหรียญอยู่จำนวนหนึ่ง) มันจะส่งเหรียญจากอีกคนไปยังอีกคน โดยมรเงื่อนไขอยู่ว่า ถ้าผู้ส่งมีเหรียญไม่พอตามจำนวนที่จะส่งก็ไม่สามารถส่งได้

2.1.4 Hierarchical Deterministic Wallet

Hierarchical Deterministic Wallet (HD Wallet) ถูกนำเสนอครั้งแรกใน BIP-32 (ย่อมาจาก Bitcoin Improvement Proposals) และต่อมา BIP-44 มาช่วยปรับปรุง BIP-32 ให้สามารถรองรับสกุลเงินดิจิทัลได้หลายสกุล สามารถสร้าง account การเก็บ Public/Private Keypairs ได้มากกว่า 1 คู่ โดยที่ใช้โครงสร้าง Tree ของมาตรฐาน BIP-32 เหมือนเดิม และ HD Wallet ถูกใช้ครั้งแรกในกลุ่มผู้ใช้ Bitcoin

โครงสร้างของ HD Wallet เป็นโครงของ Tree ดังรูปที่ 2.12 โดยมี root node ที่สร้างจากกลุ่มคำ (Seed Phrase) ตามมาตรฐาน BIP-39 ซึ่งเป็นมาตรฐานของการสร้าง Seed Phrase เพื่อนำไปใช้ใน BIP-44 ใช้กลุ่มคำนี้เป็น seed สร้าง root node (master node) ขึ้นมาสำหรับ HD Wallet นั้น และจาก root สามารถแตกต่อได้เป็นลำดับขั้น ต่อไปนี้ m / purpose / coin_type / account / chain / address_index ซึ่งค่าเริ่มต้นอีเธอเรียมจะเป็น m/44/60/0/0 ซึ่ง m หมายถึง master node , 44 คือ BIP-44 , 60 คือรหัสประเภทของเหรียญ ที่ลงทะเบียนไว้ใน SLIP-0044 , 0 ตัวต่อมาคือ Account 0 , 0 อีกตัวคือ Chain 0



รูป 2.12 โครงสร้าง Tree ของ HD Wallet

HD Wallet คือ Tree ที่มี Leaf Node เก็บ Public/Private Keypairs ซึ่งหมายความว่า 1 master node ที่สร้างจากกลุ่มคำ 1 กลุ่มสามารถมี account ได้หลาย account เช่น 1 HD Wallet สามารถเก็บเหรียญไว้ 3 ประเภทคือ BTC , ETH , EOS และแต่ละเหรียญมี account อย่างละ account และแต่ละ account มี 2 address

สิ่งที่เข้าใจผิดส่วนใหญ่ของ HD Wallet คือ master node ไม่มี keypairs ที่ใช้เซ็นรับรองธุรกรรมได้ และ ไม่สามารถใช้ account ของเหรียญคนละประเภทเซ็นรับรองธุรกรรมได้

Mnemonic Seed Phrase หรือกลุ่มคำที่ใช้สำหรับสร้าง HD Wallet ต้องเป็นคำที่มนุษย์สามารถจำได้ง่ายในภาษาต่างๆ ในกลุ่มคำตามมาตรฐาน BIP-39 กำหนดมาให้ จำนวน 2,048 คำ เช่น prevent chef demise bicycle sword network public you spider legal jazz genre

2.1.5 การออกแบบโครงสร้างของระบบสำหรับเทคโนโลยีบล็อกเชน

โครงสร้างของระบบในปัจจุบันจำเป็นต้องเขียนโปรแกรมสำหรับเซิร์ฟเวอร์ซึ่งใช้จัดกัด request ที่เข้ามา ต้องเขียนโปรแกรมเพื่อใช้สำหรับ deploy เซิร์ฟเวอร์นั้น ต้องเขียนโปรแกรมเพื่อจัดการการเข้าถึงเซิร์ฟเวอร์ และอื่นๆอีกมากมาย แต่สำหรับเครือข่ายบล็อกเชนนั้น ไม่จำเป็นทำอย่างที่กล่าวมา สำหรับการพัฒนาแอปพลิเคชันบนเครือข่ายอีเธอเรียม ต้องทำเพียงแค่เขียน smart

contract แล้ว deploy ลงบนเครือข่ายและสร้าง Client เพื่อใช้ติดต่อกับ smart contract นั้น จึงไม่จำเป็นต้องจัดการกับเซิร์ฟเวอร์

การติดต่อกับเครือข่ายอีเธอเรียมจำเป็นต้องติดต่อผ่านโหนดที่อยู่ในเครือข่าย ซึ่งอาจจะจำเป็นต้องใช้เซิร์ฟเวอร์ของตัวเองเพื่อ request จาก Client แต่ซับซ้อนเท่าระบบเดิม หรืออาจบริการ API ของ Infura เพื่อสามารถติดต่อกับโหนดได้โดยตรง สำหรับปริญญานิพนธ์นี้ผู้พัฒนาเลือกใช้บริการของ Infura เพื่อลดต้นทุนในการบริหารจัดการเครื่องเซิร์ฟเวอร์ และยังสามารถใช้บริการนี้สำหรับใช้ IPFS ซึ่งเป็นโปรโตคอลในการจัดเก็บไฟล์แบบกระจายศูนย์ (Distributed File Storage) สำหรับเครือข่ายอีเธอเรียมไม่จำเป็นต้องมีระบบจัดการกับฐานข้อมูล เพราะเทคโนโลยีบล็อกเชนคือระบบฐานข้อมูลแบบ Distributed Ledger จึงไม่มีค่าใช้จ่ายสำหรับการดูแลในส่วนนี้

2.2 งานที่เกี่ยวข้อง

2.2.1 Real estate asset tokenization in the UK

หน่วยงานด้านอสังหาริมทรัพย์ของรัฐบาลอังกฤษ Her Majesty's Land Registry (HMLR) ให้บริการลงทะเบียนอสังหาริมทรัพย์ ร่วมกับบริษัท Consensys บริษัทพัฒนาซอฟแวร์สำหรับอีเธอเรียม สร้างแนวทางในการประยุกต์ใช้เทคโนโลยีบล็อกเชนกับอุตสาหกรรมอสังหาริมทรัพย์ เพื่อแสดงข้อมูลของสินทรัพย์ไว้บนเครือข่ายบล็อกเชนในรูปแบบของ Security Token ซึ่งใช้แสดงข้อมูลในรูปแบบดิจิทัลของสินทรัพย์นั้นๆ

วิสัยทัศน์อนาคตสำหรับอุตสาหกรรมอสังหาริมทรัพย์ คือ เพื่อให้สินทรัพย์ประเภทนี้มีสภาพคล่องสูงขึ้น หาได้ง่าย ง่ายที่จะจัดการและทำเอกสาร และด้วยเทคโนโลยีบล็อกเชน ทำให้อสังหาริมทรัพย์มีความคล่องตัวในแบ่งสิทธิ์ความเป็นเจ้าของและลดค่าใช้จ่ายในการบริหารจัดการ ทำให้การบริหารจัดการและซื้อขายง่ายให้ ทำให้ตลาดของสินทรัพย์ประเภทนี้กว้างขึ้นจนถึงระดับโลก และเพิ่มความโปร่งใสในการบริหารจัดการข้อมูลของสินทรัพย์

HMLR สร้างต้นแบบของ Security Token ของสินทรัพย์อสังหาริมทรัพย์ที่มีรูปแบบเฉพาะชื่อว่า "Title Token" และถูกส่งไปยังตลาดสินทรัพย์ดิจิทัลบนเครือข่ายบล็อกเชน ของ Codefi (บริษัทลูกของ Consensys) ในตลาดนี้เจ้าของสินทรัพย์สามารถสร้าง Security Token ที่แสดงถึงส่วนแบ่งในสินทรัพย์นั้นๆ โดยเจ้าของสินทรัพย์ที่ต้องการเข้าร่วมสามารถร้องขอ Title Token ที่

แสดงสินทรัพย์ HMLR และสร้าง Security Token ขึ้นมาในตลาด โดยส่วนประกอบหลักของ Security Token ที่สร้างขึ้นมีดังนี้

1) Token name
2) Token Symbol
3) Token Supply
4) Asset Backing Token
5) Title
6) Offering Amount
7) Initial Price

เจ้าของสินทรัพย์สามารถขาย Security Token นี้ให้กับนักลงทุนในตลาดได้ และทั้งหมดมีความโปร่งใสสามารถตรวจสอบได้

2.2.2 Ripple

Ripple เป็นบริษัทที่ให้บริการโอนเงินระหว่างประเทศด้วยเทคโนโลยีบล็อกเชน ซึ่งมีความรวดเร็ว มีความน่าเชื่อถือ ครอบคลุมทุกพื้นที่ และมีค่าบริการต่ำ ผ่านเครือข่ายบล็อกเชนของตัวเองที่เรียกว่า RippleNet โดยบริการของ Ripple เน้นกลุ่มลูกค้าธนาคารจึงถูกเรียกว่า “ Blockchain for Bank ”

แอปพลิเคชัน SCB Easy ของธนาคารไทยพาณิชย์ได้ใช้บริการโอนเงินระหว่างประเทศในอัตตาค่าธรรมเนียมถูกของ Ripple ที่ชื่อว่า xCurrent โดย xCurrent เป็นระบบการส่งข้อความระหว่างธนาคาร โดยมีระบบตรวจสอบความสำเร็จของรายการโอนเงิน

และนอกเหนือการให้บริการโอนเงินระหว่างประเทศแล้ว Ripple ยังมีสกุลเงินดิจิทัลเป็นของตัวเองชื่อว่า XRP (Ripple) อีกเช่นกัน ซึ่ง XRP ถูกสร้างมาเพื่อเสริมบริการโอนเงินระหว่างประเทศ

2.2.3 Provenance

Provenance เป็น platform ที่ใช้เทคโนโลยีบล็อกเชนมาประยุกต์ใช้สำหรับการบริหารห่วงโซ่อุปทาน (Supply Chain Management) เพื่อเพิ่มความปลอดภัย ความสามารถในการตรวจสอบ Provenance สร้างตัวตนในโลกดิจิทัลให้กับสินค้าเพื่อใช้ในการยืนยันตัวตนในห่วงโซ่อุปทาน เช่น ตรวจสอบว่าใช่สินค้าตามที่บันทึกไว้หรือไม่ สินค้านี้ถูกส่งมาจากที่ไหน โดยสร้างบันทึกไว้ใน

เทคโนโลยีบล็อกเชนที่โปร่งใสและสามารถตรวจสอบได้ ซึ่งเป็นประโยชน์อย่างมากสำหรับธุรกิจค้าปลีก และการป้องกันการขายของผิดกฎหมาย

ทุกวันนี้เรารู้ข้อมูลเกี่ยวกับสิ่งค้าที่เราใช้อุปโภคบริโภคน้อยมาก ก่อนที่จะมาถึงผู้บริโภค สินค้าเหล่านี้ผ่านการขนส่ง การจัดเก็บต่าง ๆมามาก รวมถึงกระบวนการผลิต การเลือกใช้วัสดุในการผลิต ขยะที่เกิดจากการผลิต การตรวจสอบและรับรองจากองค์กรตรวจสอบภายนอก ข้อมูลเหล่านี้ถูกเก็บเป็นความลับให้เพียงแต่ผู้ที่เกี่ยวข้องรู้เท่านั้น

Provenance ต้องการให้ช่วยให้ธุรกิจค้าปลีกมีความโปร่งใส ผ่านทางข้อมูลของห่วงโซ่อุปทานที่เปิดเผยให้ผู้บริโภคได้รับรู้ ในงานวิจัย Willaim Y. and Kumju H. 2009, Sustainable consumption: green consumer behaviour when purchasing products กล่าวว่าในประเทศอังกฤษ 30% ของผู้บริโภคกังวลเกี่ยวกับปัญหาที่เกิดขึ้นต่อสิ่งแวดล้อมจากการผลิตสินค้าและมีผลต่อการตัดสินใจในการซื้อ

2.3 เครื่องมือที่ใช้งานในการพัฒนาระบบ

2.3.1 Unity

Unity คือ Cross-Platform Game Engine ซึ่งเป็นซอฟแวร์ที่ใช้สำหรับพัฒนาเกมประกอบไปด้วยเครื่องมือที่จำเป็นต่าง ๆ เช่น Graphic Engine , Audio Engine , Physics Engine , Networking , Graphical User Interface , Scripting เพื่อช่วยให้นักพัฒนาเกมสามารถพัฒนาได้ง่ายและรวดเร็ว นอกเกมแล้ว Unity ยังสามารถใช้เพื่อสร้าง Interactive Application อย่างเช่น Visual Reality ไปประยุกต์ใช้ในอุตสาหกรรมรถยนต์ สถาปัตยกรรม รวมไปถึงอุตสาหกรรมภาพยนต์สำหรับใช้สร้าง Animation

Unity สามารถไปทั้ง 2D และ 3D ทั้งขนาดเล็กและขนาดใหญ่ อีกทั้งสามารถพัฒนาโปรเจคเดียวแล้ว Export ได้ทั้ง Android, iOS, Windows, macOS, Linux, VR/AR และอื่น ๆ อีกกว่า 25 Platform โดย Unity ใช้ภาษา C# บน .Net Runtime ในการเขียน Scripting API เพื่อใช้ควบคุม Game Engine ของ Unity

Unity เป็น Game Engine ที่เป็นที่นิยมมากที่สุด มีอุปกรณ์มากกว่า 3 พันล้านเครื่องที่ติดตั้งแอปพลิเคชันที่สร้างโดย Unity และมีชุมชนผู้พัฒนาที่มีขนาดใหญ่ที่สุดในกลุ่มซอฟแวร์ประเภท Game Engine อีกทั้ง Unity ยังมี Asset Store เป็นตลาดเพื่อใช้ซื้อขาย Asset เพื่อให้ผู้พัฒนานำไปใช้

2.3.2 Nethereum

เนื่องจากเครื่องมือที่ทางผู้พัฒนาใช้ในการพัฒนาแอปพลิเคชันมือถือ คือ Unity ที่ใช้ภาษา C# บน .net Runtime และเพื่อที่ติดต่อกับเครือข่ายอีเธอเรียม เราต้องใช้โปรโตคอล JSON-RPC จึงจำเป็นต้องมีไลบราลี่ที่ช่วยในการติดต่อ

Nethereum คือไลบราลี่สำหรับเครือข่ายอีเธอเรียม ใช้สำหรับติดต่อกับ Smart Contract เช่น Deploy Smart Contract , เรียกใช้ฟังก์ชัน , ติดตามอีเวนท์ , เซ็นธุรกรรมที่ใช้ติดต่อกับเครือข่าย หรือใช้สำหรับสร้าง HD Wallet ในการใช้บริหาร Keystore หรือสำหรับ Encode argument ที่ใช้ในฟังก์ชันของ Smart Contract

2.3.3 Infura

การที่ติดต่อกับเครือข่ายอีเธอเรียมได้จำเป็นต้องติดต่อด้วยโปรโตคอล JSON-RPC กับ Full Node ซึ่งหมายความว่า จำเป็น Full Node เพื่อให้แอปพลิเคชันใช้ไลบราลี่ Nethereum ติดต่อกับ Node นี้

Infura เป็นให้บริการ API สำหรับติดต่อกับเครือข่ายอีเธอเรียมเพื่อให้ ผู้ใช้งาน API ไม่จำเป็นต้องสร้าง Node ขึ้นมาเอง โดยทาง Infura API ให้บริการ JSON-RPC โปรโตคอล ผ่านทางโปรโตคอล HTTPs และ WebSocket ผู้พัฒนาที่ใช้บริการของ Infura จะได้ประสบการณ์เหมือนใช้ Full Node ของตัวเอง นอกจาก Infura จะให้บริการ API สำหรับเครือข่ายอีเธอเรียมแล้ว Infura ยังมีบริการ API สำหรับ IPFS ด้วย

Infura ให้บริการฟรี แต่จำกัดการเรียกใช้งานที่ 100,000 ครั้งต่อวัน แต่ผู้พัฒนาสามารถซื้อขนาดการใช้งานเพิ่มเติมได้ภายหลัง โดยที่ราคาอยู่ที่ 50 ดอลล่าร์สหรัฐ ต่อเดือน จนถึง 1,000 ดอลล่าร์สหรัฐ ต่อเดือน สามารถขยายขนาดการใช้ได้เป็น 200,000 ครั้งต่อวัน จนถึง 5,000,000 ครั้งต่อวัน ผู้พัฒนาที่ใช้บริการจึงสามารถประหยัดค่าใช้จ่ายในการเปิดเซิร์ฟเวอร์เพื่อใช้สร้าง Full Node เอง

2.3.4 Truffle Suite

ในการเขียน Smart Contract ด้วยภาษา Solidity เพื่อ Deploy ลงบนเครือข่ายอีเธอเรียม จำเป็นต้องใช้ IDE เพื่อช่วยในการเขียน คอมไพล์ และ Deploy ลงบนเครือข่าย IDE สำหรับภาษา Solidity อย่างเช่น Remix IDE ที่ทำงานบนเว็บบราวเซอร์

Truffle Suite คือ ชุดเครื่องมือที่ช่วยพัฒนาแอปพลิเคชันบนเครือข่ายอีเธอเรียม ประกอบไปด้วย

1) Truffle เป็น framework ที่ช่วยในการพัฒนา Smart Contract บนเครือข่ายอีเธอเรียมช่วยในการคอมไพล์ภาษา Solidity, Linking, Deploy และ Automated Testing ของ Smart Contract

2) Ganache เป็นโปรแกรมที่ใช้สร้างเครือข่ายอีเธอเรียมจำลองสำหรับการพัฒนา

3) Drizzle คือ Frontend library ที่ช่วยในการสร้างเว็บแอปพลิเคชัน

2.3.5 IPFS

จากหลักการของบล็อกเชนที่ว่า ระบบไม่จำเป็นต้องมีศูนย์กลางในการเก็บข้อมูล เช่นเดียวกันกับระบบที่ใช้เก็บไฟล์ IPFS ถูกสร้างมาเพื่อทดแทนระบบแบบรวมศูนย์ที่จำเป็นต้องเครื่องเซิร์ฟเวอร์เพื่อใช้บริการไฟล์ผ่านเครือข่ายอินเตอร์เน็ต ซึ่งมีค่าใช้จ่ายสูง

Inter-Planetary File System หรือ IPFS คือโปรโตคอลสำหรับระบบจัดการและจัดเก็บไฟล์แบบกระจายศูนย์ มีจุดประสงค์เพื่อทดแทนโปรโตคอล HTTP ในอนาคต ทุก ๆวันนี้อินเตอร์เน็ตแบบรวมศูนย์ไม่มีประสิทธิภาพ เพราะ HTTP ดาวน์โหลดไฟล์จากเครื่องเซิร์ฟเวอร์เพียงเครื่องเดียวทำให้เซิร์ฟเวอร์เครื่องรับ Bandwidth จำนวนมาก แต่ IPFS ให้บริการไฟล์คอมพิวเตอร์หลายเครื่องพร้อมกันแบบ Peer-to-Peer จึงทำให้ประหยัด Bandwidth มากกว่าระบบแบบรวมศูนย์ในปัจจุบัน

บทที่ 3

การออกแบบและพัฒนา

3.1 ขั้นตอนการดำเนินงาน

ขั้นตอนการจัดทำปริญญานิพนธ์ แบ่งออกเป็น ดังนี้

1) ศึกษาเทคโนโลยีที่เกี่ยวข้อง

2) กำหนดปัญหาและขอบเขตของปริญญานิพนธ์

3) ออกแบบการทำงานของแอปพลิเคชัน

4) ออกแบบโครงสร้างของระบบ

5) สร้างแอปพลิเคชันตามการออกแบบ

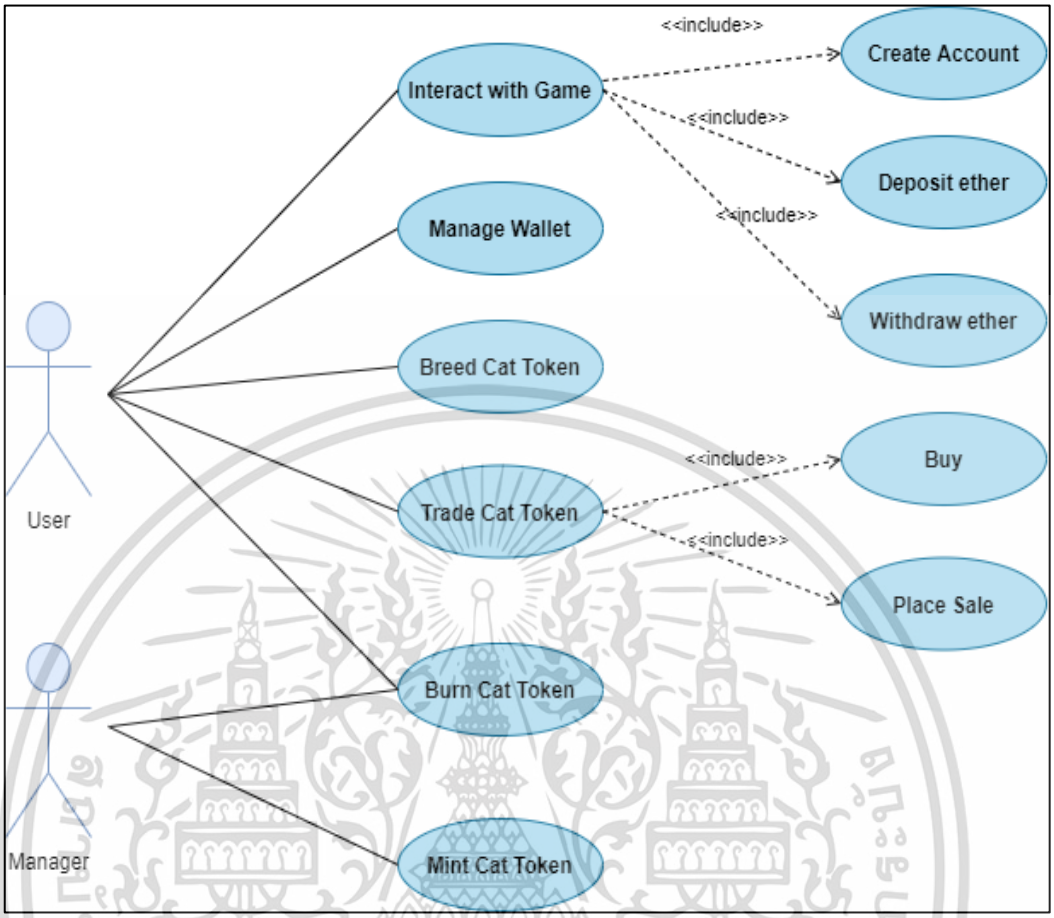
6) ทดสอบคุณภาพของแอปพลิเคชัน

7) สรุปผล

3.2 การออกแบบการทำงาน

3.2.1 Use Case Diagram

จุดประสงค์ของแอปพลิเคชันนี้คือให้ผู้ใช้ได้เรียนรู้และเข้าถึงการใช้งานจริงของเทคโนโลยีบล็อกเชน ในรูปแบบของ Interactive Game ภายในแอปพลิเคชันผู้ใช้จะได้เป็นเจ้าของสินทรัพย์ดิจิทัลที่ชื่อว่า Cat Token ที่อยู่บนเครือข่ายอีเธอเรียมและผู้ใช้ยังสามารถซื้อขาย และส่งต่อ Cat Token นี้ไปให้ผู้อื่นได้ เพราะผู้ใช้ได้รับความเป็นเจ้าของโดยสมบูรณ์ของสินทรัพย์นี้ ดัง Use Case Diagram ที่แสดงในรูปที่ 3.1



รูป 3.1 Use case Diagram

รายละเอียดของของแต่ละ Use Case มีดังนี้

1) Manage Wallet คือการจัดการกับกระเป๋าเงินของเครือข่ายอีเธอเรียม ได้แก่ 1. การสร้างบัญชีเพื่อใช้เก็บเหรียญ Ether (ETH) ฝากเหรียญ ถอนเหรียญ และใช้สำหรับจัดการคู่กุญแจ (Key Pair) เพื่อสร้าง transaction ติดต่อกับ smart contract

2) Interact with Game คือการเล่นเกมเลี้ยงแมวเพื่อให้สีสันความสนุกให้กับผู้ใช้ ได้แก่ ให้อาหารแมว อาบน้ำให้แมว ทำความสะอาดห้อง เล่นกับแมว โดยการกระทำเหล่านี้จะเพิ่มคะแนนความสนิทของเรากับแมว ซึ่งจะมีผลในการขายหรือเพราะพันธุ์แมว

3) Breed Cat Token คือ การนำแมว 2 ตัวมาผสมพันธุ์กัน โดยนำยีนของแมวทั้งสองมาผสมกัน และส่งต่อไปยังลูก แล้วให้กำเนิดลูกออกมาเป็นเหรียญใหม่บนเครือข่าย

4) Trade Cat Token คือ ซื้อขายแลกเปลี่ยนเหรียญแมวดิจิทัลในตลาดสำหรับการซื้อขาย ผู้ใช้จะสามารถตั้งประกาศขายแมว และสามาเข้าซื้อแมวที่ตั้งประกาศขายไว้ได้

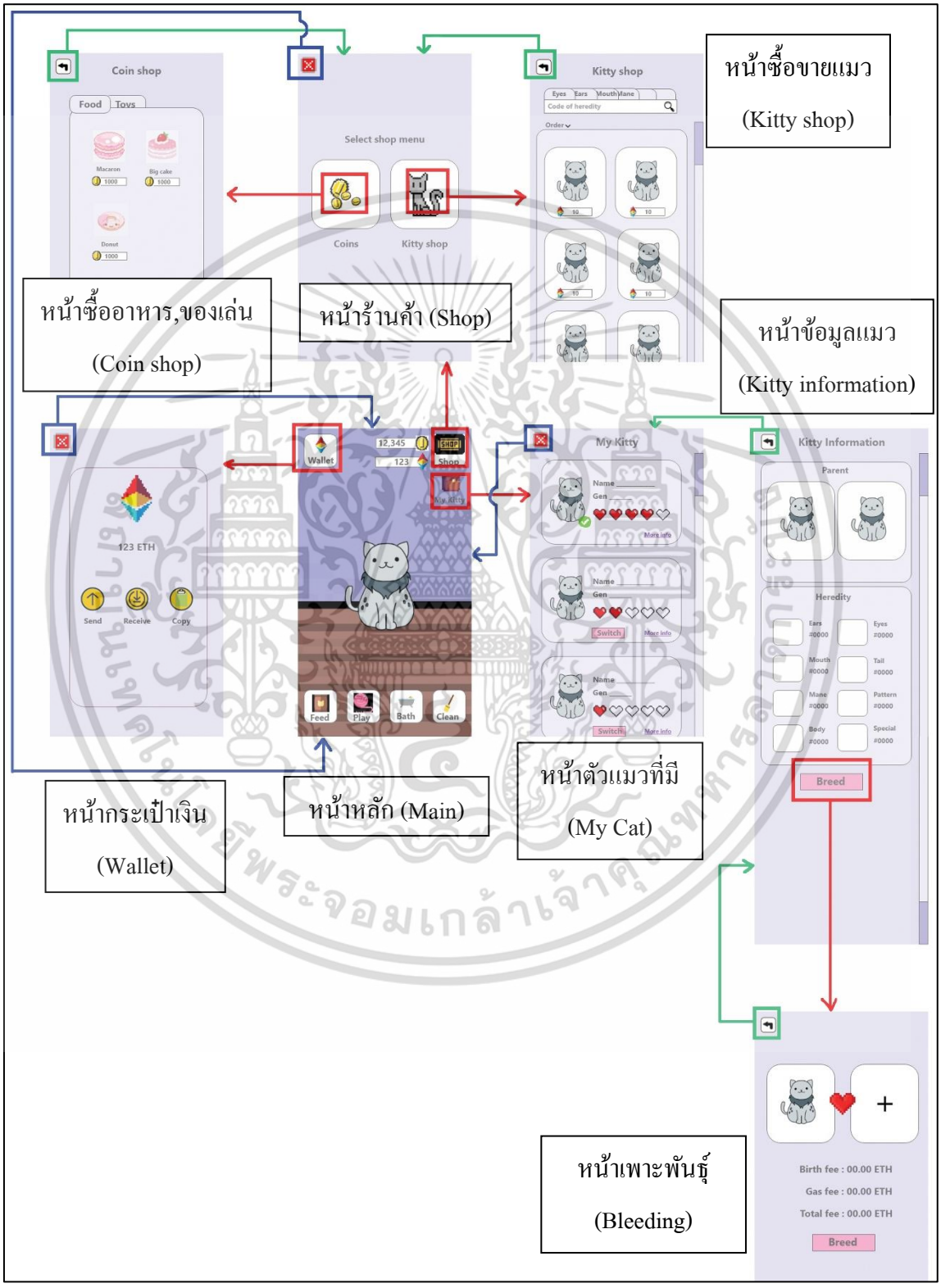
5) Transfer Cat Token คือ ผู้ใช้สามารถส่งเหรียญแมวดิจิทัลไปให้ผู้ใช้คนอื่นได้

6) Mint Cat Token คือ สร้างเหรียญแมวดิจิทัลขึ้นมาโดยที่ไม่ต้องมีการผสมพันธุ์ ซึ่งต้องทำโดยผู้ที่ได้รับอนุญาตเท่านั้น

7) Burn Cat Token คือ ทำลายเหรียญแมวดิจิทัลทิ้ง ซึ่งต้องทำโดยผู้ที่ได้รับอนุญาตและเจ้าของเหรียญเท่านั้น

3.3 การออกแบบส่วนติดต่อผู้ใช้งาน (User Interface)

3.3.1 ภาพรวมของส่วนติดต่อผู้ใช้งาน



รูป 3.2 User Interface Overview

3.4 การออกแบบโครงสร้างของระบบ

3.4.1 System Architecture

โครงสร้างของระบบในโครงการนี้แบ่งออกเป็น 3 ส่วน ได้แก่

1) Client Mobile (Android)

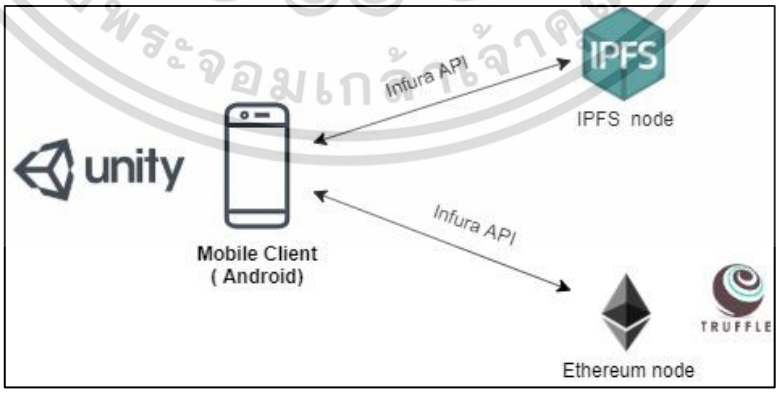
2) Ethereum Node Logic และ Database ของระบบ

3) IPFS Node

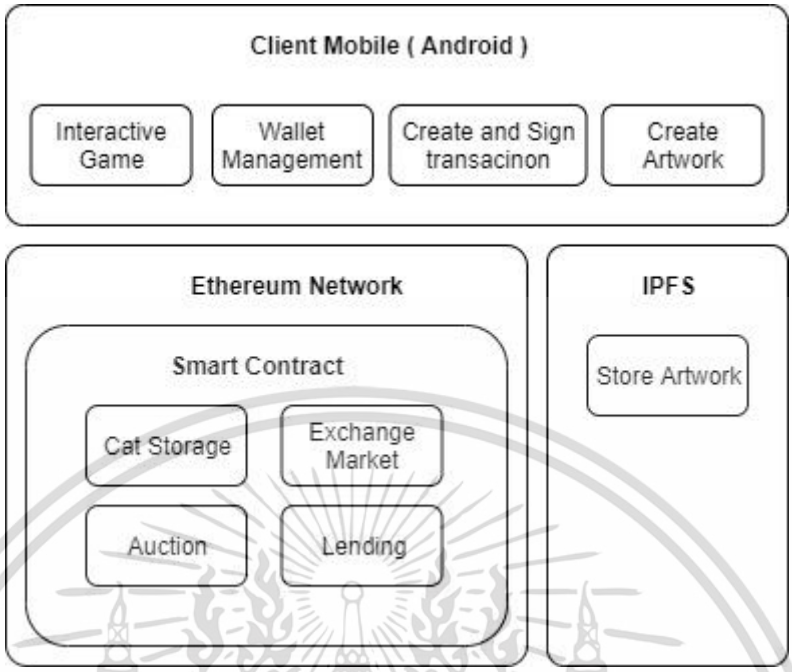
Client Mobile คือส่วนติดต่อกับผู้ใช้งาน ใช้สำหรับควบคุมและแสดงผลส่วนผู้ใช้งาน ใช้สำหรับบริหารและจัดเก็บคู่กุญแจของผู้ใช้งาน ใช้สำหรับสร้าง transaction เพื่อใช้ติดต่อกับเครือข่ายอีเธอเรียม ใช้สำหรับเซ็น transaction ด้วยคู่กุญแจที่เก็บไว้แล้วส่งไปยังเครือข่ายอีเธอเรียม และใช้สำหรับสร้าง artwork ของแมวดิจิทัลเพื่อไปเก็บไว้ใน IPFS และในส่วนของ Client Mobile นี้ใช้ Unity ในการพัฒนาซึ่งสามารถ build ลงอุปกรณ์ android ได้

Ethereum Node คือโหนดในเครือข่ายอีเธอเรียม ซึ่งติดต่อกับ Client Mobile ด้วยบริการของ Infura ซึ่งไว้สำหรับ Deploy Smart Contract ที่เขียนด้วยภาษา Solidity ซึ่ง Smart Contract ที่สร้างขึ้นเปรียบเหมือนกับส่วน Business Logic และ Database ของระบบ สำหรับเครื่องที่ใช้เขียน deploy และ test smart contract คือ truffle และ ganache ในชุด truffle suite

IPFS Node ใช้สำหรับเก็บไฟล์รูป Artwork ของแมวดิจิทัลแต่ละตัวซึ่งถูกสร้างและอัพโหลดโดยตรงจาก Client Mobile

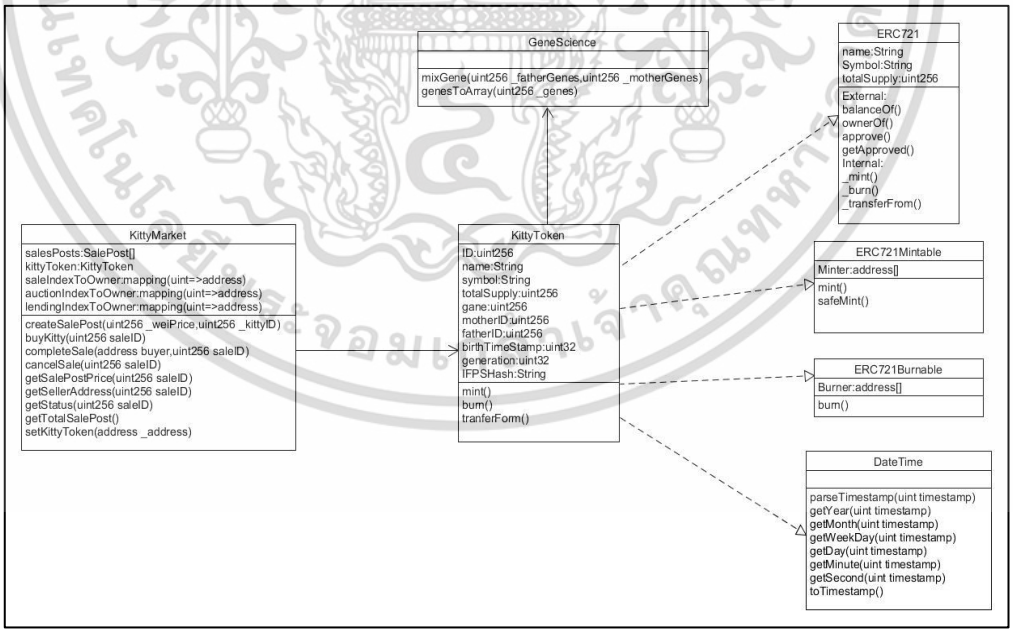


รูป 3.3 แสดงโครงสร้างความสัมพันธ์ระหว่างองค์ประกอบของระบบ



รูป 3.4 แสดงโครงสร้างแบบลำดับชั้นของโครงสร้างระบบ

3.4.2 Class diagram



รูป 3.5 Class Diagram ของ Smart Contract ทั้งหมด

ในรูปที่ 3.5 Smart contract ประกอบไปด้วย 7 class ได้แก่

1) KittyToken เป็น contract หลักในการเก็บข้อมูลเหรียญดิจิทัล และสร้างเหรียญดิจิทัล

2) ERC721 เป็น contract จากไลบราลี่ openzepelinContract เป็นคลาสตามมาตรฐาน ERC721 ของอีเธอเรียม เพื่อใช้สร้างสินทรัพย์ดิจิทัลที่ไม่สามารถแบ่งแยกต่อได้

3) ERC721Mintable เป็น contract จากไลบราลี่ openzepelinContract เพื่อเสริม ERC721 contract ให้สามารถ mint หรือสร้างสินทรัพย์ดิจิทัลนั้นขึ้นมา

4) ERC721Burnable เป็น contract จากไลบราลี่ openzepelinContract เพื่อเสริม ERC721 contract ให้สามารถ burn หรือทำลายสินทรัพย์ดิจิทัลที่มีอยู่ทิ้ง

5) KittyMarket เป็น contract ที่ใช้สำหรับตั้งประกาศขายหรือซื้อแมวดิจิทัล

6) GeneScience เป็น contract เพื่อผสมยีนหรือรหัสพันธุกรรมของแมวดิจิทัลที่เป็นประเภท uint256 2 ค่ามาผสมกันแล้วเกิดเป็นยีนใหม่

7) DateTime เป็น contract ที่ใช้สำหรับการจัดการกับวันที่และเวลาที่ใช้ในเครือข่ายอีเธอเรียม

3.4.3 รายละเอียดของคลาสต่างๆ

3.4.3.1 KittyToken

รูป 3.6 คลาสของแมวดิจิทัล

ข้อมูลของแมวดิจิทัลนี้ประกอบไปด้วย

1) ตัวแปรประเภท uint256 ชื่อ ID แสดงลำดับหมายเลข

2) ตัวแปรประเภท string ชื่อ name แสดงชื่อของแมว

3) ตัวแปรประเภท string ชื่อ symbol แสดงสัญลักษณ์โดยย่อ ซึ่งแสดงเหมือนกันทุกตัว

4) ตัวแปรประเภท uint256 ชื่อ totalSupply แสดงจำนวนแมวทั้งหมดที่มีอยู่

5) ตัวแปรประเภท uint256 ชื่อ gene แสดงลักษณะของแมวที่ปรากฏออกมา

6) ตัวแปรประเภท uint256 ชื่อ motherID แสดง ID ของแม่

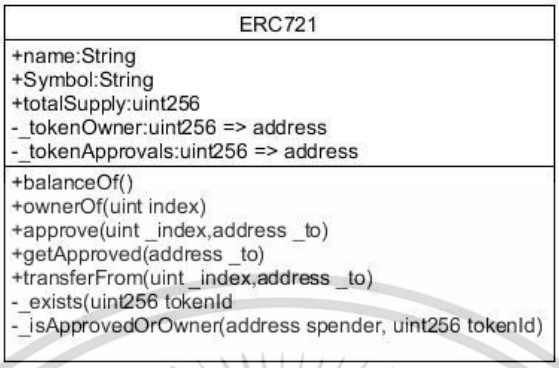
7) ตัวแปรประเภท uint256 ชื่อ fatherID แสดง ID ของพ่อ

8) ตัวแปรประเภท uint256 ชื่อ birthTimeStamp แสดง UNIX Time Stamp ณ เวลาที่แมวเกิด

9) ตัวแปรประเภท uint256 ชื่อ generation แสดง รุ่นของแมว

10) ตัวแปรประเภท string ชื่อ IPFSHash แสดง hash ของ IPFS ที่ใช้เก็บไฟล์รูปภาพของแมว

3.4.3.2 ERC721

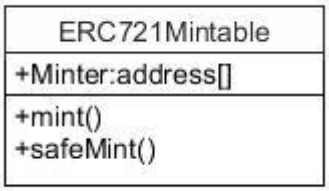


รูป 3.7 ERC721 class diagram

ใน contract ERC721 นี้ใช้เพื่อสร้างสินทรัพย์ดิจิทัลที่ไม่สามารถแบ่งแยกได้ ตามมาตรฐาน ERC 721 ของอีเธอเรียม มี attribute 3 อย่างคือ 1) Name ชื่อของสินทรัพย์ดิจิทัลกลุ่มนี้ 2) Symbol สัญลักษณ์โดยย่อของสินทรัพย์ดิจิทัลนั้น 3) totalSupply จำนวนสินทรัพย์ดิจิทัลที่อยู่ในกลุ่มนี้ทั้งหมด

และยังประกอบด้วยฟังก์ชันอีก 5 ฟังก์ชัน คือ 1.balanceOf() ใช้เพื่อตรวจสอบจำนวนสินทรัพย์ดิจิทัลกลุ่มนี้ที่ถือครองอยู่ 2.ownerOf ใช้เพื่อตรวจสอบว่าใครเป็นเจ้าของสินทรัพย์ดิจิทัลชิ้นนี้ 3.approve ใช้เพื่อมอบอำนาจในการจัดการสินทรัพย์ดิจิทัลชิ้นนั้นให้ผู้อื่น 4.getApproved ใช้เพื่อตรวจสอบว่ามีอำนาจในการจัดการสินทรัพย์ดิจิทัลนั้นหรือไม่ 5.transferFrom ใช้เพื่อส่งสินทรัพย์ดิจิทัลที่ครอบครองอยู่หรือมีอำนาจในการจัดการไปให้ผู้อื่น

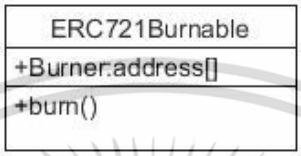
3.4.3.3 ERC721Mintable



รูป 3.8 ERC721Mintable class diagram

contract นี้มาจากไลบรารี่ openzepelinContract เพื่อเสริม ERC721 contract ให้สามารถ mint หรือสร้างสินทรัพย์ดิจิทัลนั้นขึ้นมา มี attribute 1 อย่างคือ Minter หรือผู้ที่มีอำนาจในการ mint มีฟังก์ชัน mint

3.4.3.4 ERC721Burnable



รูป 3.9 ERC721Burnable class diagram

contract นี้มาจากไลบรารี่ openzepelinContract เพื่อเสริม ERC721 contract ให้สามารถ burn หรือทำลายสินทรัพย์ดิจิทัลนั้นทิ้ง มี attribute 1 อย่างคือ Burner หรือผู้ที่มีอำนาจในการ burn มีฟังก์ชัน burn

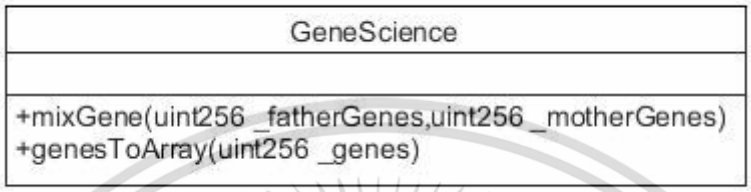
3.4.3.5 KittyMarket

รูป 3.10 KittyMarket class diagram

KittyMarket เป็น contract ที่ใช้สำหรับตั้งประกาศขายหรือซื้อแมวดิจิทัล contract นี้มีอำนาจในการจัดการสินทรัพย์ของผู้ที่ประกาศขาย มี attribute 3 อย่างคือ 1.salePosts เป็นอา

เรย์ของ salePost ที่ใช้เก็บข้อมูลการประกาศขายแมวดิจิทัล 2.kittyToken เก็บ address ของ KittyToken Contract เอาไว้เพื่อใช้สื่อสารกัน 3. saleIndexToOwner ใช้เพื่อเชื่อมประกาศขายเข้ากับผู้ที่ประกาศขาย

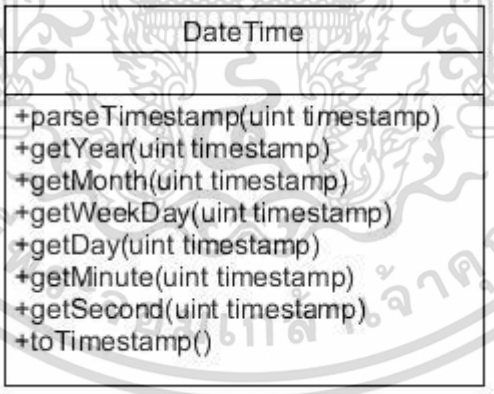
3.4.3.6 GeneScience



รูป 3.11 GeneScience class diagram

GeneScience เป็น contract ที่ใช้ผสมยีนให้กับ KittyToken เมื่อเกิดการผสมพันธุ์กัน มีฟังก์ชัน 2 ฟังก์ชัน คือ 1. MixGene() ใช้ผสมยีนจากพ่อและแม่ออกมาเป็นยีนใหม่ 2.geneToArray() แปลงค่ายีนที่อยู่ในรูป uint256 ให้เป็นอาเรย์ที่เก็บตัวแปรประเภท int ลักษณะทั้งหมด 48 ลักษณะ

3.4.3.7 DateTime



รูป 3.12 DateTime class diagram

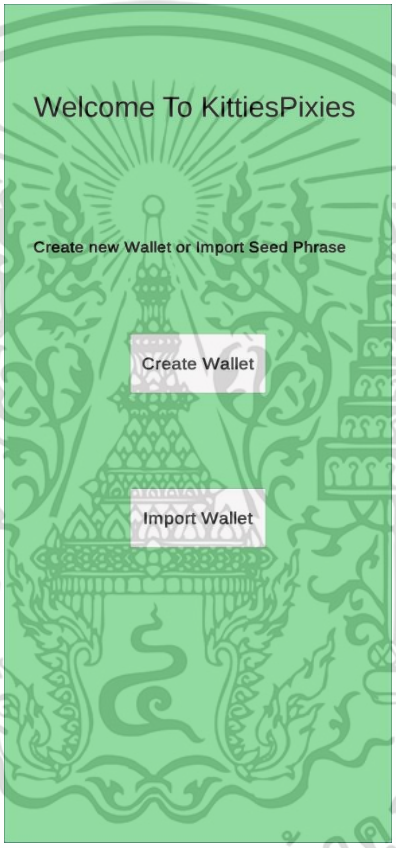
ใช้เพื่อแปลง UNIX Timestamp ให้อยู่ในรูปของ DateTime

บทที่ 4

การทดลองและผลการทดลอง

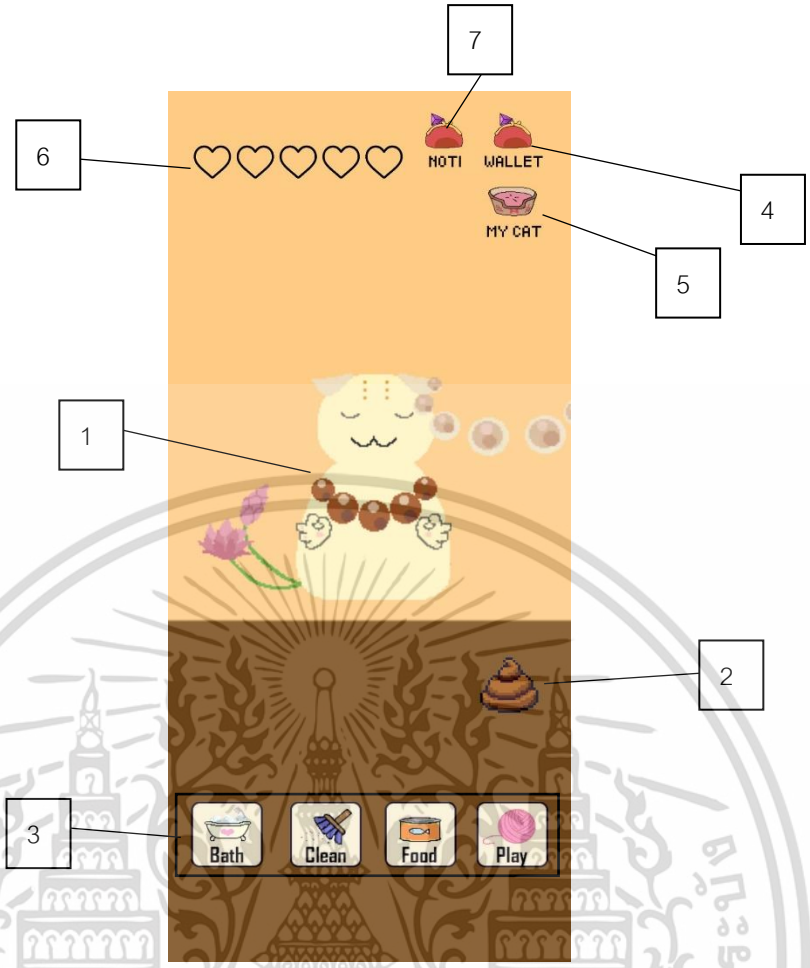
4.1 ส่วนแสดงผลทางหน้าแอปพลิเคชัน

4.1.1 หน้าต่าง Log in



รูป 4.1 หน้าต่าง Log in

เมื่อผู้ใช้งานเข้าใช้งานแอปพลิเคชันเป็นครั้งแรก จะพบกับหน้า Log in นี้เพื่อสร้าง Wallet สำหรับใช้งานในแอปพลิเคชันหรือสามารถนำเข้า Wallet ที่มีอยู่แล้วด้วยชุดคำ 12 คำ (seed phrase) จากนั้นจึงสามารถเข้าใช้งานแอปพลิเคชันได้



รูป 4.2 หน้าต่าง main Screen

4.1.2 หน้าต่าง Main Screen

ที่หน้าต่างหลักจะแสดงผลดังรูป 4.2 ซึ่งจะมีองค์ประกอบ 7 ส่วนดังนี้

1) รูปตัวแมว เป็นสินทรัพย์ดิจิทัล

2) ไอคอนสถานะผิดปกติ ซึ่งเป็นสภาวะที่ทำให้เราสามารถใช้งานปุ่มเลี้ยงแมวตามอาการเพื่อเพิ่มค่าหัวใจได้

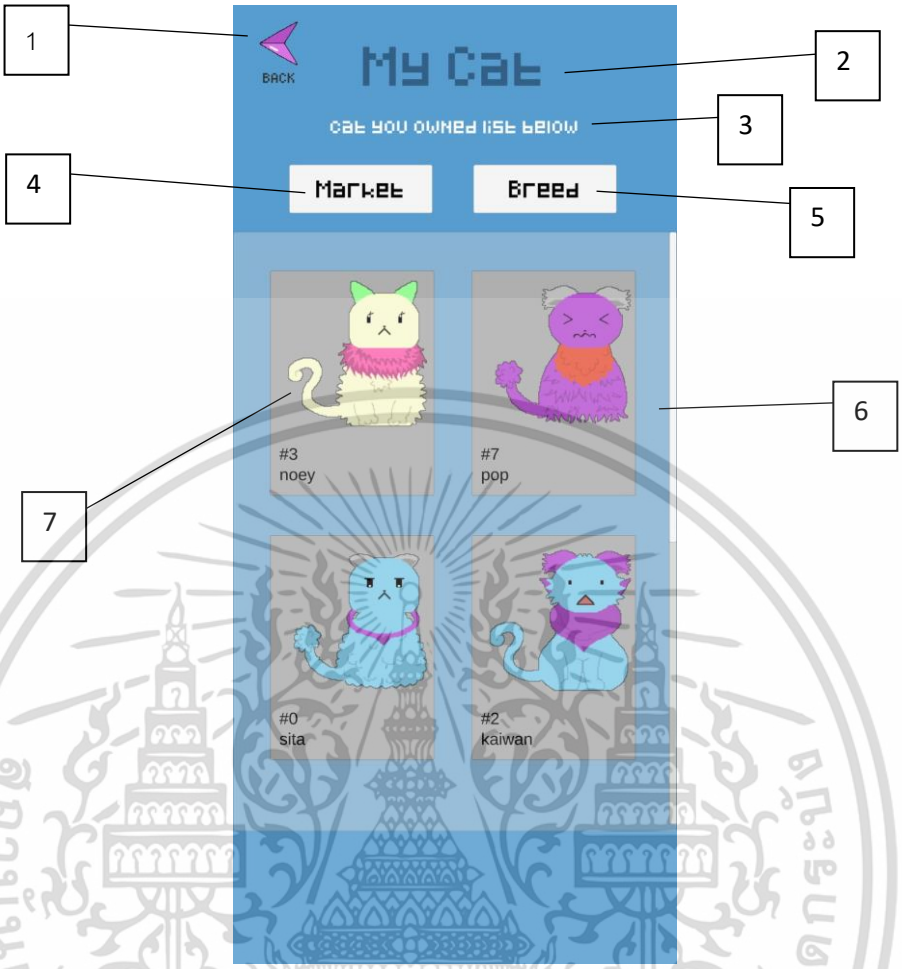
3) ปุ่มเลี้ยงแมว เป็นปุ่มที่ไว้ลบล้างสถานะผิดปกติตามอาการ

4) ปุ่ม WALLET คือปุ่มที่ไว้แสดงเงิน ETH ของเรา

5) ปุ่ม MY CAT คือคลังตัวแมวที่เรามีทั้งหมดรวมถึงร้านค้า

6) ไอคอนหัวใจ มีไว้แสดงค่าความรักของตัวแมวที่เราเลี้ยงอยู่ ซึ่งค่าความรักมีผลต่อดูลดาวน์ในการผสมพันธุ์

7) ปุ่ม NOTI คือปุ่มที่กดเพื่อไปยังหน้า Transaction Notification ดังหัวข้อ 4.1.9



รูป 4.3 หน้าต่าง My Cat

4.1.3 หน้าต่าง My Cats

หน้าต่างนี้ไว้เพื่อแสดงแมวที่ผู้เล่นครอบครองไว้

ที่หน้าต่าง My Cats จะแสดงผลดังรูป 4.3 ซึ่งจะมีองค์ประกอบ 7 ส่วนดังนี้

1) ปุ่ม Back มีไว้เพื่อกลับไปยังหน้าต่างหลัก

2) ชื่อของหน้าต่าง My Cats

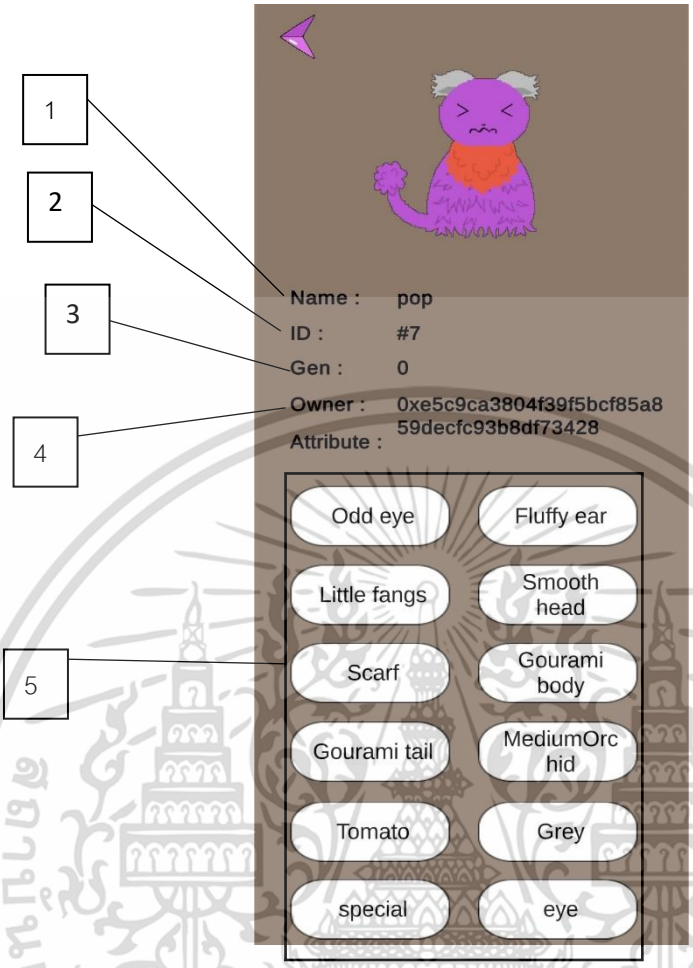
3) คำอธิบายของหน้าต่าง My Cats บอกว่า แมวที่คุณครอบครองแสดงอยู่ด้านล่าง

4) ปุ่ม Market คือปุ่มที่กดเพื่อไปยังหน้า Market ดังหัวข้อ 4.1.6

5) ปุ่ม Breed คือปุ่มที่กดเพื่อไปยังหน้า Breeding Cat ดังหัวข้อ 4.1.8

6) Scroll View เป็นบริเวณที่ใช้แสดงรายการแมวที่ครอบครอง

7) แสดงข้อมูลโดยย่อของแมว ประกอบด้วย รูปของแมว, ไอดี และชื่อตามลำดับ



รูป 4.4 หน้าต่าง Cat Detail

4.1.4 หน้าต่าง Cat Detail

หน้านี้แสดงรายละเอียดแมวตัวที่ถูกเลือกโดยแสดงรายละเอียด ดังนี้

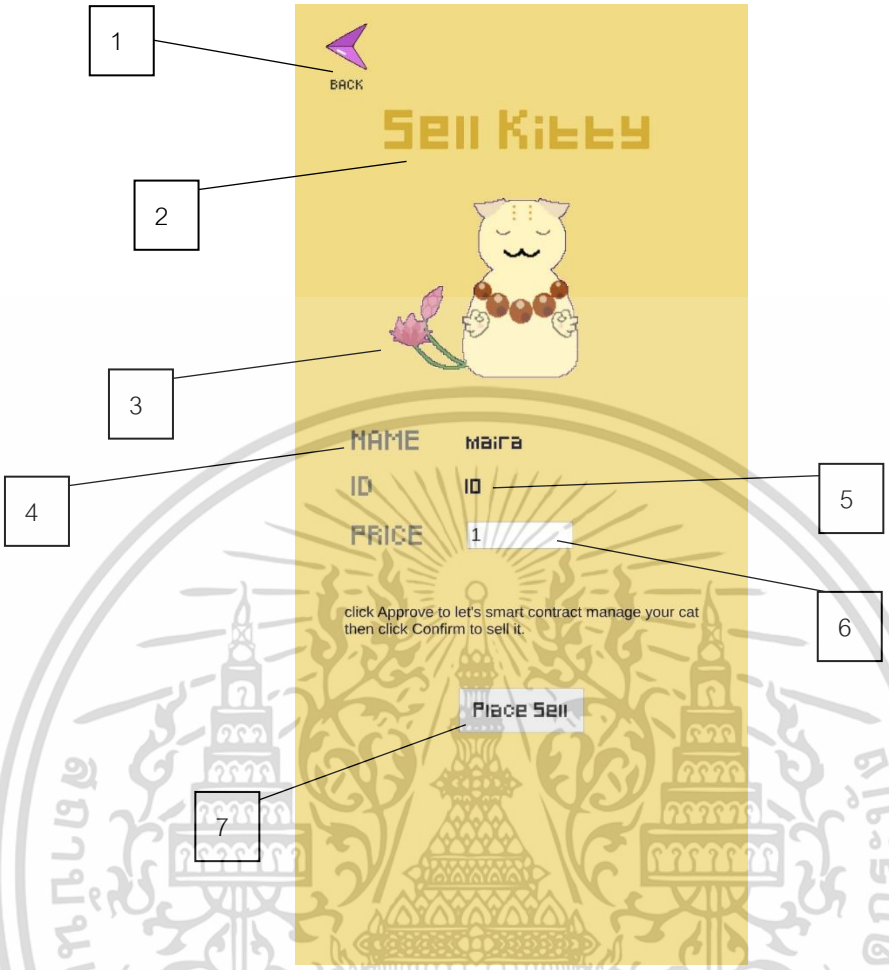
1. ชื่อของตัวแมว (Name)

2. ไอดี (ID)

3. รุ่น (Generation)

4. รหัสประจำตัวเจ้าของ (Owner's address)

5. ลักษณะที่ปรากฏ (attribute) โดยที่ด้านล่างสุดของหน้านี้มีปุ่ม Sell ใช้เพื่อกดไปยังหน้า Selling ในข้อที่ 4.1.4 เพื่อตั้งประกาศขายแมวตัวนี้



รูป 4.5 หน้าต่าง Selling

4.1.5 หน้าต่าง Selling

หน้านี้ผู้เล่นสามารถตั้งประกาศขายแมวของตัวที่เลือกไว้ได้ที่หน้านี้

ที่หน้าต่าง Selling จะแสดงผลดังรูป 4.5 ซึ่งจะมีองค์ประกอบ 7 ส่วนดังนี้

1) ปุ่ม Back มีไว้เพื่อกลับไปยังหน้าต่าง Market

2) ชื่อหน้าต่าง Sell Kitty

3) รูปแมวที่ต้องการประกาศขาย

4) ชื่อของแมวที่ต้องการประกาศขาย

5) ไอดีของแมวที่ต้องการประกาศขาย

6) ช่องไว้เพื่อกรอกราคาของแมวที่ต้องการประกาศขายหน่วยเงินเป็น ETH

7) ปุ่ม Place Sell กดเพื่อประกาศขาย

รูป 4.6 หน้าต่าง Market

4.1.6 หน้าต่าง Market

หน้าต่างนี้มีไว้เพื่อซื้อตัวแมวดิจิทัลที่ถูกประกาศขายไว้

ที่หน้าต่าง Market จะแสดงผลดังรูป 4.6 ซึ่งจะมีองค์ประกอบ 5 ส่วนดังนี้

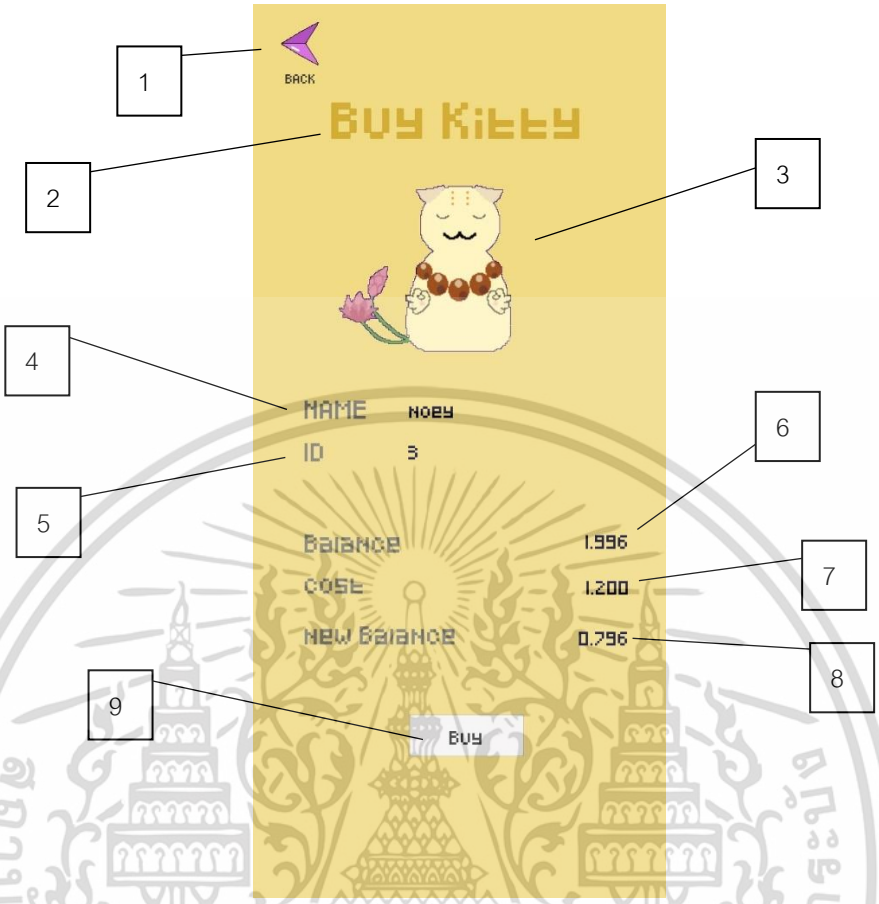
1) ปุ่ม Back ใช้เพื่อกลับไปยังหน้าต่าง My Cats

2) ชื่อของหน้าต่าง Market

3) แถบตัวกรอง (filter) ใช้กรองลักษณะต่างๆของแมวที่วางขาย ประกอบด้วย Attribute เลือกลักษณะของแมว , Attribute Name เลือกชื่อของลักษณะ , Sort by เลือกตัวแปรในการเรียงผล และสุดท้ายลำดับการเรียงจากมากไปน้อย หรือน้อยไปมาก

4) Scroll View บริเวณที่ใช้แสดงผลแมวที่ถูกประกาศขาย

5) รายละเอียดโดยย่อของแมว ประกอบด้วย รูป, ไอดี, ชื่อ และราคา ตามลำดับ



รูป 4.7 หน้าต่าง Buying

4.1.7 หน้าต่าง Buying

หน้านี้ผู้เล่นสามารถซื้อแมวที่ประกาศขายไว้ในหน้า Market

ที่หน้าต่าง Buying จะแสดงผลดังรูป 4.7 ซึ่งจะมีองค์ประกอบ 9 ส่วนดังนี้

1) ปุ่ม Back ใช้เพื่อกลับไปยังหน้าต่าง Market

2) ชื่อของหน้าต่าง Buying

3) รูปของแมวที่ต้องการซื้อ

4) ชื่อของแมวที่ต้องการซื้อ

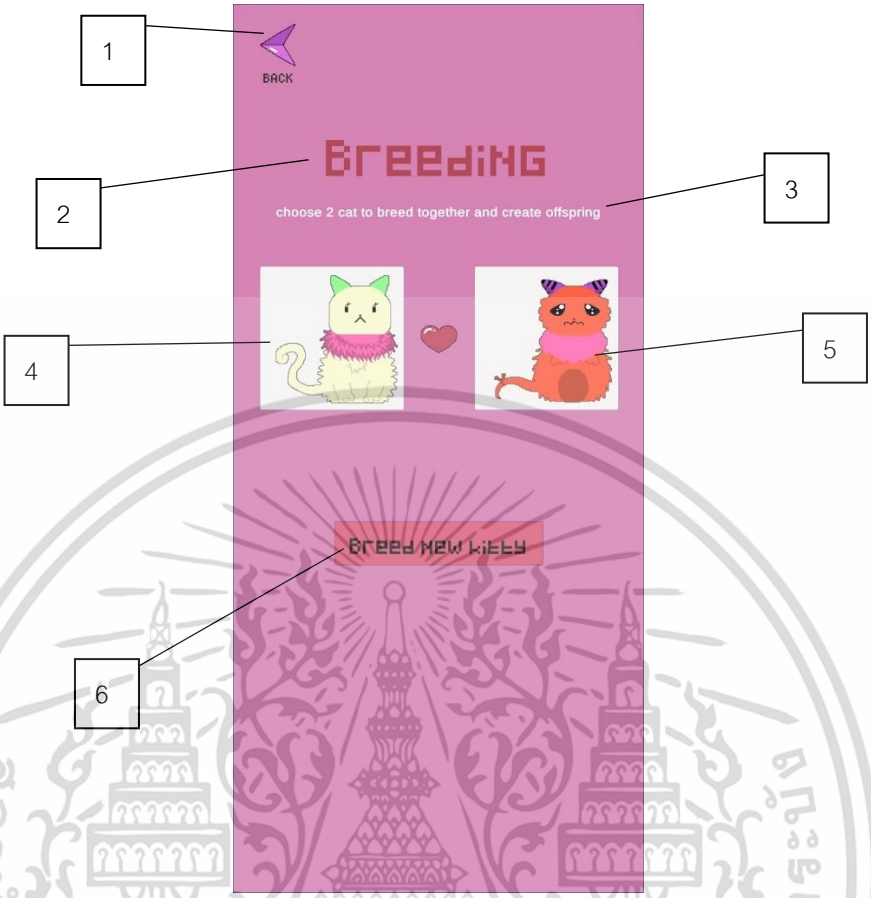
5) ไอดีของแมวที่ต้องการซื้อ

6) ยอดเงินในหน่วย ETH ของ Account ปัจจุบัน

7) ราคาของแมวที่ต้องการซื้อ

8) ยอดเงินที่จะเหลือหลังจากซื้อแมว

9) ปุ่ม Buy เพื่อส่งคำสั่งซื้อแมวตัวนี้



รูป 4.8 หน้าต่าง Breeding Cat

4.1.8 หน้าต่าง Breeding Cat

ใช้ในการผสมพันธุ์ตัวแมว 2 ตัวเพื่อให้ได้ตัวแมวตัวใหม่ ระหว่างดำเนินการผสมพันธุ์จะให้ผู้ใช้งานรอเป็นเวลา 30 นาทีแล้วจึงกลับเข้ามาตรวจสอบอีกครั้งภายหลัง ซึ่งหากผสมพันธุ์สำเร็จตัวแมวที่เกิดใหม่จะเป็นสินทรัพย์ของเราทันที

ที่หน้าต่าง Breeding Cats จะแสดงผลดังรูป 4.8 ซึ่งจะมีองค์ประกอบ 6 ส่วนดังนี้

1) ปุ่ม Back ใช้เพื่อกลับไปยังหน้าต่าง My Cats

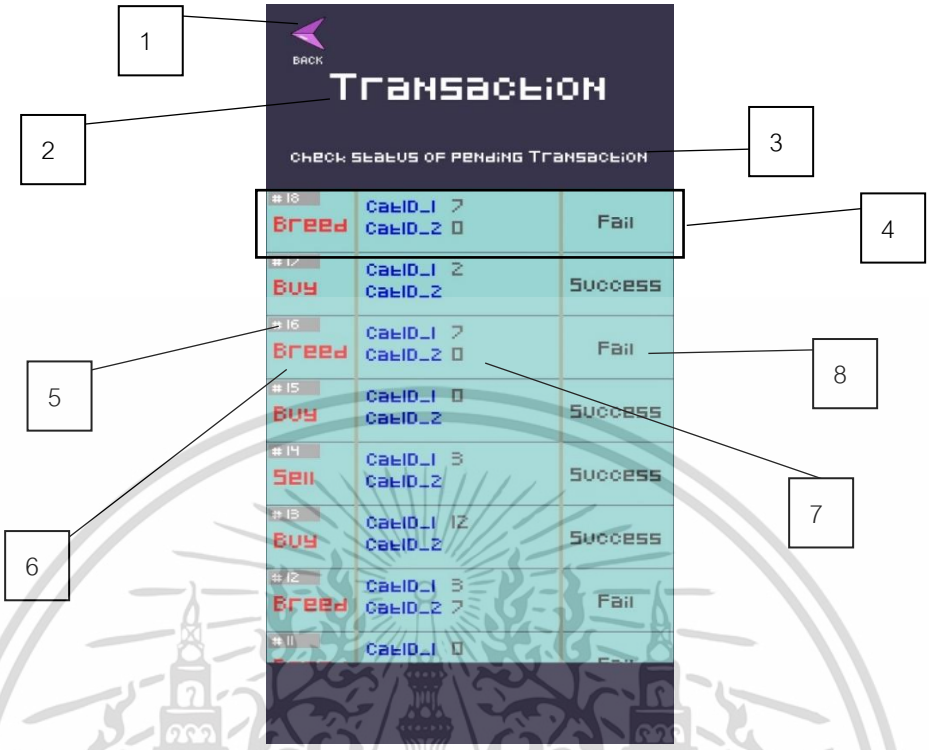
2) ชื่อของหน้าต่าง Breeding Cat

3) คำอธิบายของหน้าต่างนี้

4) ปุ่มเลือกแมวตัวที่ 1 ที่มาใช้ในการผสมพันธุ์

5) ปุ่มเลือกแมวตัวที่ 2 ที่มาใช้ในการผสมพันธุ์

6) ปุ่ม Breed New Kitty เพื่อส่งคำสั่งเพาะพันธุ์แมว



รูป 4.9 หน้าต่าง Transaction Notification

4.1.9 หน้าต่าง Transaction Notification

เนื่องจากการยืนยันใช้เวลานาน อาจมากกว่า 15 วินาที ซึ่งส่งผลเสียต่อประสบการณ์การใช้งานที่ต้องให้ผู้ใช้รอเป็นเวลานาน เพราะฉะนั้นเมื่อใดก็ตามที่ที่มีการสร้าง transaction เกิดขึ้น แอปพลิเคชันจะให้ผู้ใช้กลับไปยังหน้าหลักเพื่อรอการยืนยัน และสามารถตรวจสอบ Transaction นั้นได้ที่หน้านี้

ที่หน้าต่าง Transaction Notification จะแสดงผลดังรูป 4.9 ซึ่งจะมีองค์ประกอบ 8 ส่วนดังนี้

1) ปุ่ม Back ใช้เพื่อกลับไปยังหน้าต่างหลัก

2) ชื่อของหน้าต่าง Transaction Notification

3) คำอธิบายสำหรับหน้าต่างนี้

4) รายละเอียดของ Transaction ล่าสุดที่ผู้เล่นสร้างขึ้นมา

5) เลขลำดับของ Transaction

6) ชื่อฟังก์ชันที่ใช้ใน Transaction

7) ไอดีของแมว 1 หรือ 2 ตัว ที่ใช้ในฟังก์ชันนั้น

8) สถานะของ Transaction (Pending , Success หรือ Fail)

4.2 การติดตั้งและตั้งค่าเครื่องมือที่ใช้ในการพัฒนา

จำเป็นต้องติดตั้งเครื่องมือที่จำเป็นก่อนเริ่มการพัฒนา

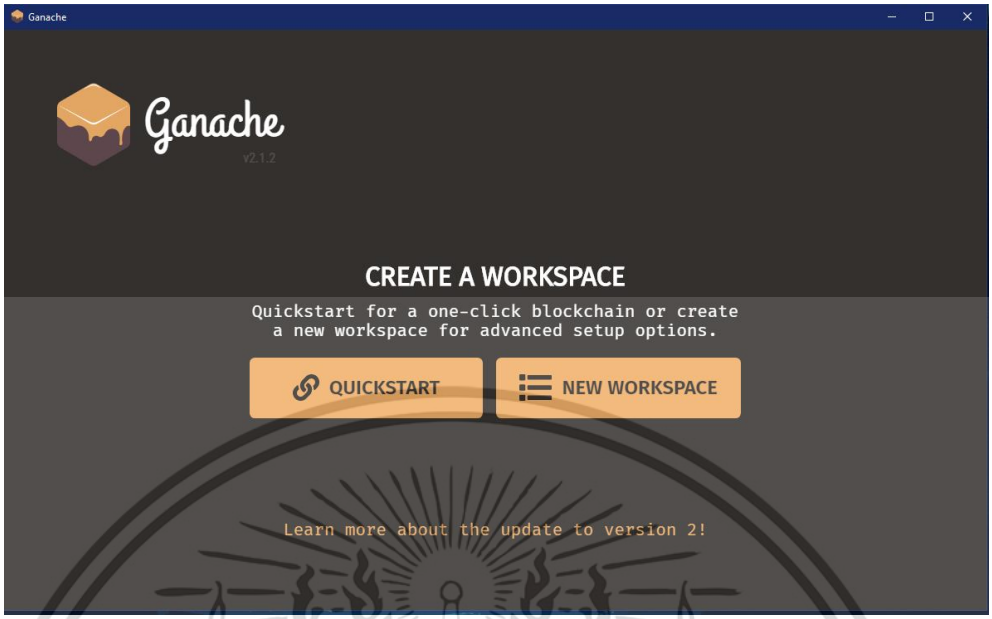
4.2.1 การติดตั้ง Ganache

Ganache คือเครือข่ายบล็อกเชนทดลอง ที่สามารถสร้างเครือข่ายที่ใช้สำหรับทดลอง Ethereum Smart Contract ลงบนเครือข่ายที่อยู่เครื่องคอมพิวเตอร์ของผู้พัฒนาได้โดยตรง โดยที่ไม่ต้องไปติดต่อกับเครื่องคอมพิวเตอร์อื่น สามารถตั้งค่าได้ตามสะดวก

เริ่มแรกดาวน์โหลดตัวติดตั้งผ่านเว็บไซต์ https://www.trufflesuite.com/ganache เมื่อดาวน์โหลดเสร็จ ทำการติดตั้งด้วยตัวติดตั้ง ดังรูปที่ 4.10

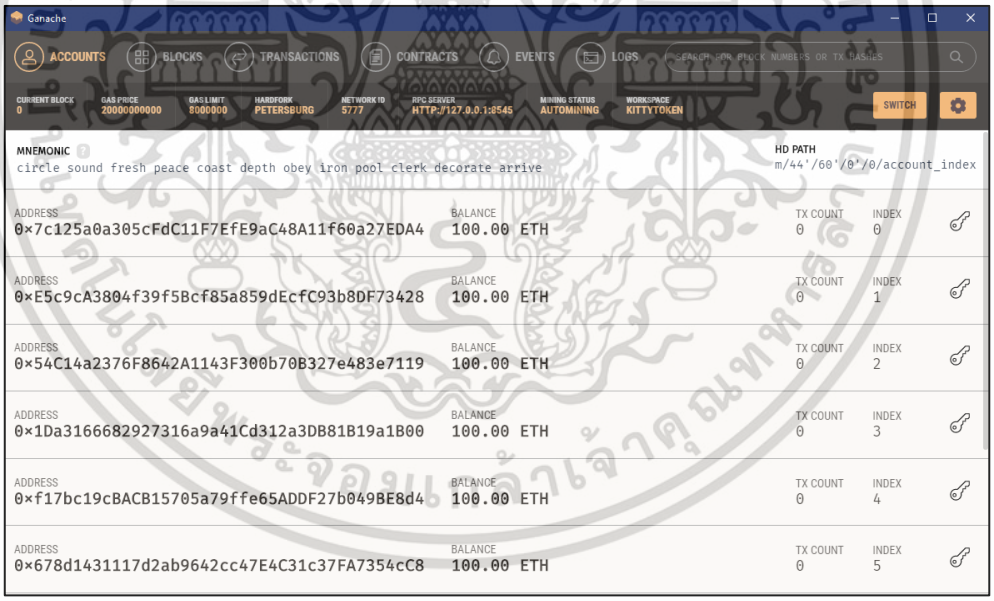
รูป 4.10 ติดตั้ง Ganache

เมื่อติดตั้งสำเร็จ มันจะให้เลือกระหว่าง Quickstart (เริ่มสร้าง Workspace ขึ้นทันที) กับ New-Workspace (สร้าง workspace แบบตั้งค่าเอง) ดังรูปที่ 4.11



รูป 4.11 สร้าง Ganache Workspace

เมื่อสร้าง workspace ขึ้นมาสำเร็จแล้วจึงได้เครือข่ายอีเธอเรียมทดลองพร้อมใช้งานดังรูปที่ 4.12



รูป 4.12 หน้าต่าง Ganache Workspace

4.2.2 การติดตั้ง Node และ Truffle

Truffle คือ framework ที่ใช้สำหรับพัฒนา Ethereum Smart Contract ช่วยจัดการ Compile ภาษา Solidity , Deploy Smart Contract , Debug Smart Contract และ Automated Smart Contract

Test ด้วย moและ chai เนื่องจาก truffle สร้างขึ้นบน Node.js จึงจำเป็นต้องดาวน์โหลดและติดตั้ง Node.js ก่อน แล้วใช้คำสั่งเพื่อติดตั้ง truffle เป็น Global Dependencies ด้วย NPM (node Package Manager)

```
$ npm install -g truffle
```

เมื่อติดตั้ง truffle เสร็จแล้วจึงสร้าง truffle project ชื่อ KittiesSmartContract ขึ้นมาด้วยคำสั่ง

```
$ mkdir KittiesSmartContract
$ cd KittiesSmartContract
$ truffle init
```

เมื่อสร้าง project ขึ้นมาเสร็จในโฟลเดอร์ KittiesSmartContract ประกอบไปด้วย

1) /contracts คือโฟลเดอร์ที่ใช้เก็บ Smart Contract ภาษา Solidity

2) /migrations คือโฟลเดอร์ที่ใช้เก็บโค้ดภาษา Javascript สำหรับ deploy Smart Contract

3) /test คือโฟลเดอร์ที่ใช้เก็บโค้ดสำหรับ Automated Test ด้วย macha หรือ chai

4) truffle-config.js คือ ไฟล์ที่ใช้สำหรับตั้งค่า truffle project

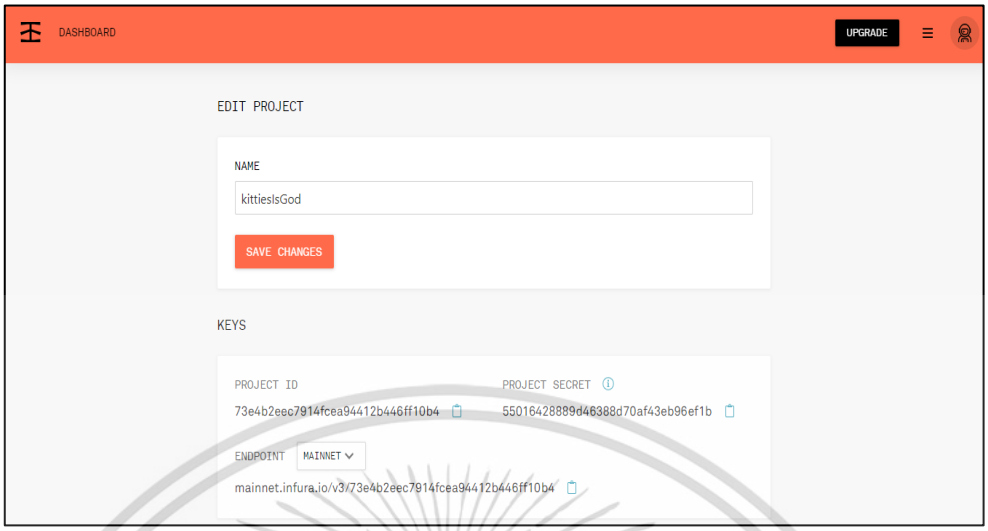
4.2.3 การสมัครใช้งานบริการ API ของ Infura

Infura คือผู้ให้บริการ API เพื่อใช้ติดต่อกับ Ethereum Node ด้วยโปรโตคอล HTTP อีกทั้งยังมี API ที่ใช้สำหรับเชื่อมต่อ IPFS Node อีกด้วย การใช้จะ API ได้จำเป็นต้องมี API Key ดังนั้นจึงต้องสมัครสมาชิกก่อนผ่านเว็บไซต์ https://infura.io ดังรูปที่ 4.13

รูป 4.13 สมัครสมาชิก Infura

เมื่อสมัครเสร็จแล้วก็สร้าง Project เพื่อใช้งาน API ดังรูปที่ 4.14 และ 4.15

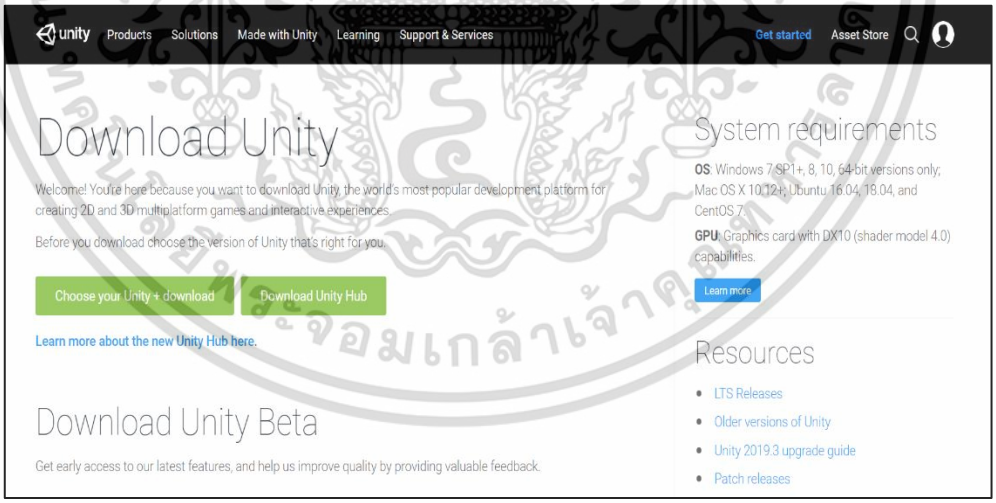
รูป 4.14 สร้าง infura project



รูป 4.15 ได้ API key เพื่อเข้าใช้งาน

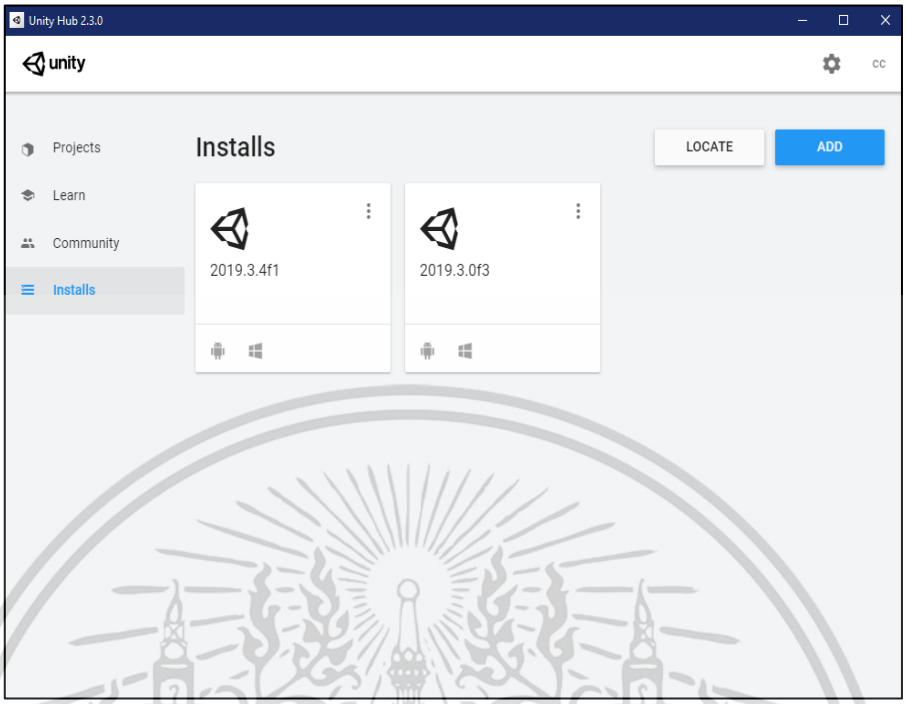
4.2.4 การติดตั้ง Unity

Unity คือ Cross Platform Game Engine ที่สามาใช้พัฒนาเกมลงบน Android ได้ เริ่มต้นด้วยการดาวน์โหลด Unity เวอร์ชั่นล่าสุดจากเว็บไซต์ https://unity3d.com/get-unity/download ดาวน์โหลด unity hub ดังรูปที่ 4.16 และทำการติดตั้ง



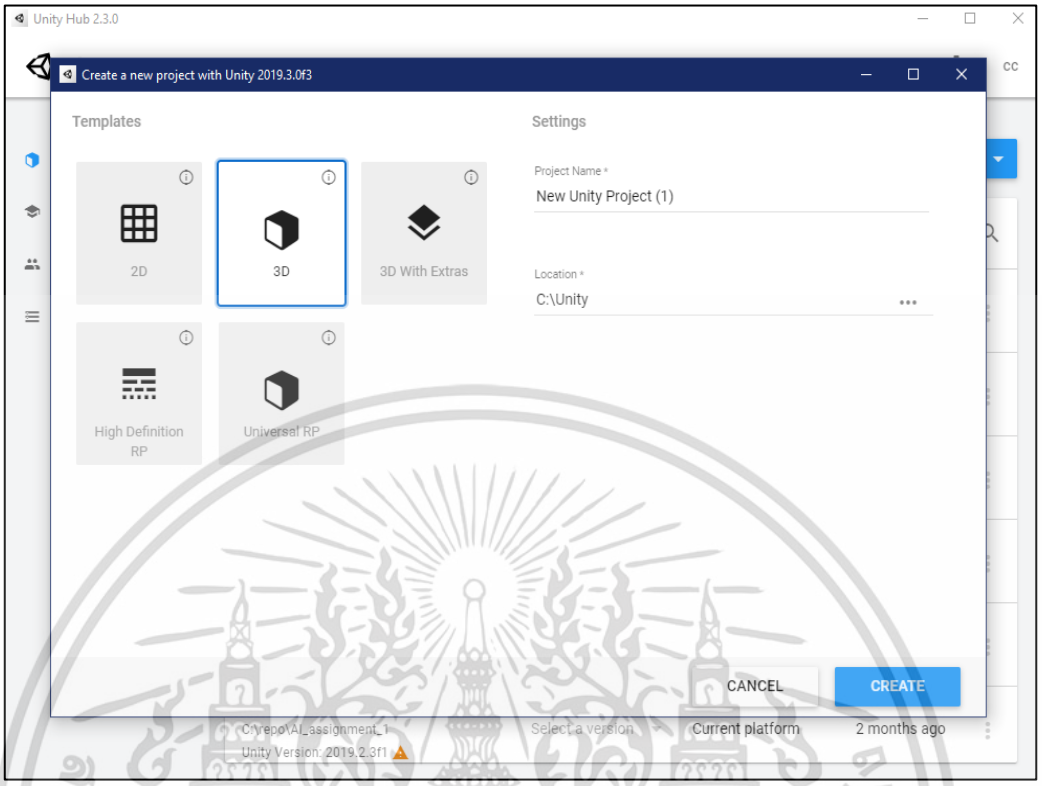
รูป 4.16 ดาวน์โหลด Unity

เมื่อติดตั้งเสร็จเรียบร้อยแล้วใช้ Unity Hub ติดตั้ง Unity เวอร์ชันล่าสุดที่ Installs tab ด้านซ้ายดังรูปที่ 4.17



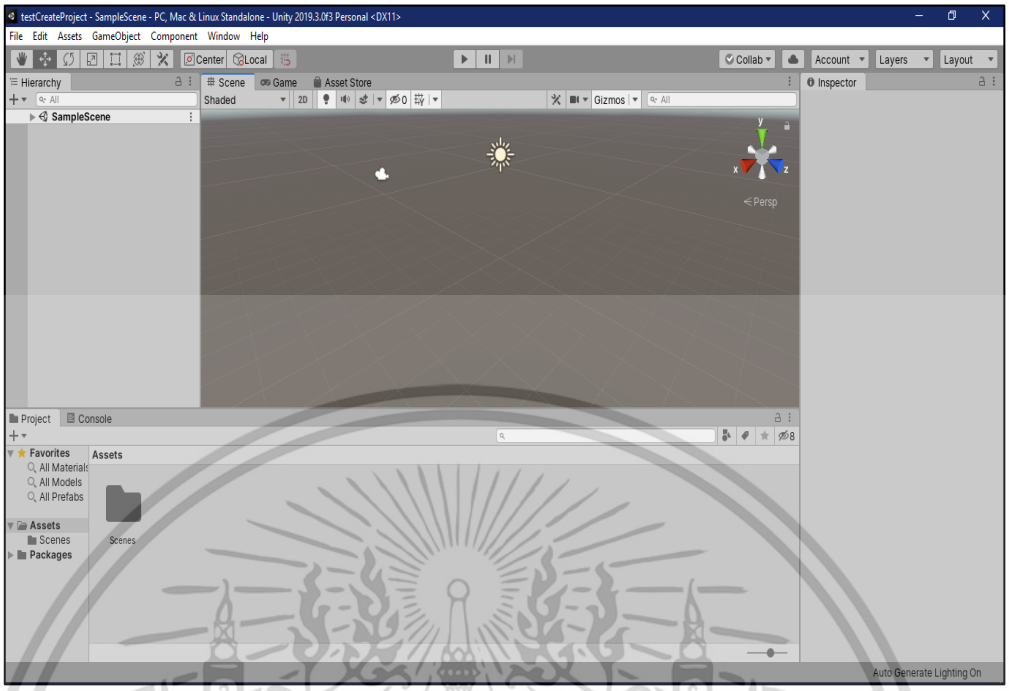
รูป 4.17ติดตั้ง Unity เวอร์ชันล่าสุด ผ่าน Unity Hub

เมื่อติดตั้ง Unity สำเร็จแล้ว จากนั้นสร้าง Project ขึ้นดังรูปที่ 4.18 ในหน้าสร้าง Project มีให้เลือก template ที่จะใช้ใน Project เลือก 2D



รูป 4.18 สร้าง Unity Project ใหม่

เมื่อสร้าง Unity Project สำเร็จจะได้ผลลัพธ์ดังรูปที่ 4.19 บน Unity Editor ประกอบไปด้วย 4 ส่วนหลัก คือ 1. ด้านซ้าย Hierarchy แสดงรายการ Game Object ที่อยู่ใน scene ปัจจุบัน 2. ตรงกลาง Scene/Game แสดงโลกหรือฉากที่เราทำงานในปัจจุบัน 3. ด้านขวา Inspector แสดงข้อมูลรายละเอียดของ Game Object ที่เราเลือก 4. ด้านล่าง Project แสดงโฟลเดอร์และไฟล์ของ project นี้



รูป 4.19 หน้าต่างเริ่มต้น Unity

4.3 การสร้างและทดลอง Smart Contract ด้วย Truffle

4.3.1 การนำเข้าไลบรารี่ ชื่อ OpenZeppelin และสร้าง ERC721 Smart Contract

OpenZeppelin เป็นสำหรับใช้พัฒนา Smart Contract ช่วยให้ผู้พัฒนาสร้างแอปพลิเคชันบนบล็อกเชนได้สะดวกขึ้น เช่น OpenZeppelin มี Interface ERC20 (มาตรฐานที่ใช้สำหรับสร้าง Crypto Currency บนอีเธอเรียม) และ ERC721 (มาตรฐานสำหรับสร้างของสะสมที่ไม่สามารถแบ่งได้) ให้ใช้ Implement ใน Contract ของผู้พัฒนา นอกจากนี้ยังช่วยในเรื่องของ การจำกัดการเข้าถึง (Access Control), ความปลอดภัย (Security) และอื่นๆ

ในปริญญานิพนธ์ทางผู้พัฒนาใช้ไลบรารี่ OpenZeppelin นี้ในการสร้างเหรียญที่ไม่สามารถแบ่งแยกต่อได้ ชื่อว่า KittyToken ด้วยการ implement Interface ERC721 ก่อนอื่นจึงต้องเริ่มจากการติดตั้งไลบรารี่นี้ลงในเครื่องของผู้พัฒนา ด้วยขั้นตอนจากเว็บไซต์

https://github.com/OpenZeppelin/openzeppelin-contracts

เริ่มแรกที่โฟลเดอร์ KittiesSmartContract ที่เป็น truffle project ที่ถูกสร้างขึ้นมาก่อนหน้านี้ และทำการติดตั้งไลบรารี่ด้วยคำสั่ง

```
$ npm install @openzeppelin/contracts
```

จากนั้นสร้าง contract ชื่อว่า KittyToken แล้ว implement Interface ของ openzeppelin ดังรูปที่ 4.20

```
1  pragma solidity >=0.5.0;
2
3  import "@openzeppelin/contracts/token/ERC721/ERC721Full.sol";
4  import "@openzeppelin/contracts/token/ERC721/ERC721Mintable.sol";
5
6  contract KittyToken is ERC721Full, ERC721Mintable {
7      constructor() ERC721Full("DeizyKittyToken", "DKTN") public {
8      }
9  }
```

รูป 4.20 เริ่ม implement ERC721

เมื่อ Implement Interface ERC721 แล้ว contract นี้สามารถใช้ฟังก์ชัน mint หรือสร้างเหรียญขึ้นมาได้ แต่เหรียญที่สร้างขึ้นมานั้นยังไม่มีคุณสมบัติที่ทำให้เหรียญนั้นแตกต่างกัน จึงจำเป็นต้องสร้างตัวแปร struct ที่ชื่อว่า Kitty และอาเรย์ ชื่อ kitties เพื่อเก็บข้อมูลของเหรียญ ดังรูปที่ 4.20 ซึ่ง struct Kitty นี้สร้างขึ้นมาเพื่อใช้เก็บข้อมูลเหรียญแมวดิจิทัล ประกอบไปด้วย 1. nameOfKitty ตัวแปรประเภท string คือ ชื่อของแมวดิจิทัล 2. Genes ตัวแปรประเภท uint256 คือ ยีนของแมวดิจิทัล ใช้เพื่อบ่งบอกลักษณะของแมวตัวนั้น ๆ และยังใช้คำนวณเพื่อผสมพันธุ์ระหว่างแมวดิจิทัล 3. fatherID ตัวแปรเภท uint256 คือ ไอดีของพ่อของแมวตัวนั้น ๆ 4. motherID ตัวแปรประเภท uint256 คือ ไอดีของแม่ของแมวตัวนั้น ๆ 4. birthTimeStamp ตัวแปรประเภท DateTime คือ วันที่และเวลาที่แมวตัวนั้น ๆ ถูกสร้างขึ้นมา 5. Generation ตัวแปรประเภท uint256 คือ รุ่นของแมวตัวนั้น ๆ หาได้จากรุ่นของพ่อหรือแม่ที่มีค่ามากสุดบวก 1 6. IPFSHash ตัวแปรประเภท string คือ ค่า hash ของเครือข่าย IPFS ที่ใช้เก็บรูป Artwok ของแมวตัวนั้น ๆ

```
struct Kitty {
  string nameOfKiity;
  uint256 genes;
  uint256 fatherID;
  uint256 motherID;
  _DateTime birthTimeStamp;
  uint256 generation;
  string IPFSHash;
}

Kitty[] public kitties;
```

รูป 4.21 โครงสร้าง struct และ array เพื่อเก็บข้อมูลของเหรียญ

จากนั้นสร้างฟังก์ชันเพื่อสร้างเหรียญขึ้นมาสำหรับใช้ภายใน contract ชื่อว่า mintKitty ในฟังก์ชันนี้จะสร้าง kitty struct ขึ้นมาตาม parameter ที่รับมาแล้วนำไปใส่ในอาเรย์ ดังรูปที่ 4.22

```
function mintKitty(
    address _to,
    string memory _nameOfKiity,
    uint256 _fatherID,
    uint256 _motherID,
    uint256 _generation,
    uint256 _genes,
    string memory _IPFSHash
  ) internal returns (uint256){

  uint256 newItemId = _tokenIds.current();

  Kitty memory _kitty = Kitty({
    nameOfKiity:_nameOfKiity,
    genes:_genes,
    fatherID:_fatherID,
    motherID:_motherID,
    generation:_generation,
    birthTimeStamp:parseTimestamp(now),
    IPFSHash:_IPFSHash
  });

  kitties.push(_kitty); //push new kitty to array
  _mint(_to, newItemId); //ERC721 mint function
  _tokenIds.increment(); //kitty ID running number increase

  emit Birth(_to,newItemId,_fatherID,_motherID,_genes);

  return newItemId;
}
```

รูป 4.22 ฟังก์ชัน mintKitty

ต่อไปสร้างฟังก์ชันเพื่อให้เรียกใช้ฟังก์ชัน mintKitty จากไปภายนอกได้ ชื่อว่า createPromoKitty ดังรูปที่ 4.23 ฟังก์ชันนี้อนุญาตเฉพาะเจ้าของ contract นี้หรือเรียกว่า CEO เท่านั้น ที่สามารถเรียกใช้ฟังก์ชันนี้ได้ โดยที่สามารถสร้างแมวรูปแบบได้ก็ได้ขึ้นมา

```
function createPromoKitty(
  address _to,
  string memory _nameOfKiity,
  uint256 _fatherID,
  uint256 _motherID,
  uint256 _generation,
  uint256 _genes,
  string memory _IPFSHash
)
public
onlyMinter
returns (uint256){
  return mintKitty(_to,_nameOfKiity,_fatherID,_motherID,_generation,_genes,_IPFSHash);
}
```

รูป 4.23 ฟังก์ชัน createPromoKitty

เมื่อเขียน Smart Contract ขึ้นมาเรียบร้อยแล้ว ต่อไปต้อง deploy ลงบน Ganache เพื่อที่จะได้ทดสอบฟังก์ชันที่สร้างขึ้นมา เริ่มสร้างไฟล์ชื่อ 2_deploy_KittyToken.js ขึ้นโฟลเดอร์ /migrations ใน truffle project โดยที่เลข 2 ในชื่อไฟล์คือเลขลำดับการ Deploy และเขียนโค้ดดังรูปที่ 4.24 จากนั้นใช้คำสั่ง $ truffle migrate

```
var KittyToken = artifacts.require("KittyToken");

module.exports = function(deployer) {
  // deployment steps
  deployer.deploy(KittyToken);
};
```

รูป 4.24 deploy KittyToken

4.3.2 การทดสอบ KittyToken Contract ผ่าน Truffle Console

Truffle มี console สำหรับใช้ติดต่อกับ Smart Contract ที่อยู่บนเครือข่ายไหนก็ได้ (ทั้ง local development network , test network และ main network) โดยที่ไม่ต้องติดตั้งไลบรารีอื่นเพิ่มอีก เริ่มแรกเปิด command prompt บน windows ขึ้นมา ไปที่ truffle project แล้วใช้คำสั่ง

```
$ truffle console
```

จากนั้นจะเข้าสู่ truffle console

```
truffle(development) >
```

สร้าง contract instance ขึ้นบน console ด้วยคำสั่ง ซึ่ง 0x3836F74dE3CE532146237b75193c6C5053E712eF คือ address ของ contract ที่ deploy ไปก่อนหน้านี้

```
truffle console(development) > let kitty = await
KittyToken.at("0x3836F74dE3CE532146237b75193c6C5053E712eF")
```

จากนี้สามารถส่ง transaction เพื่อเรียกใช้ฟังก์ชัน createPromoKitty บน smart contract ได้ด้วยคำสั่งนี้

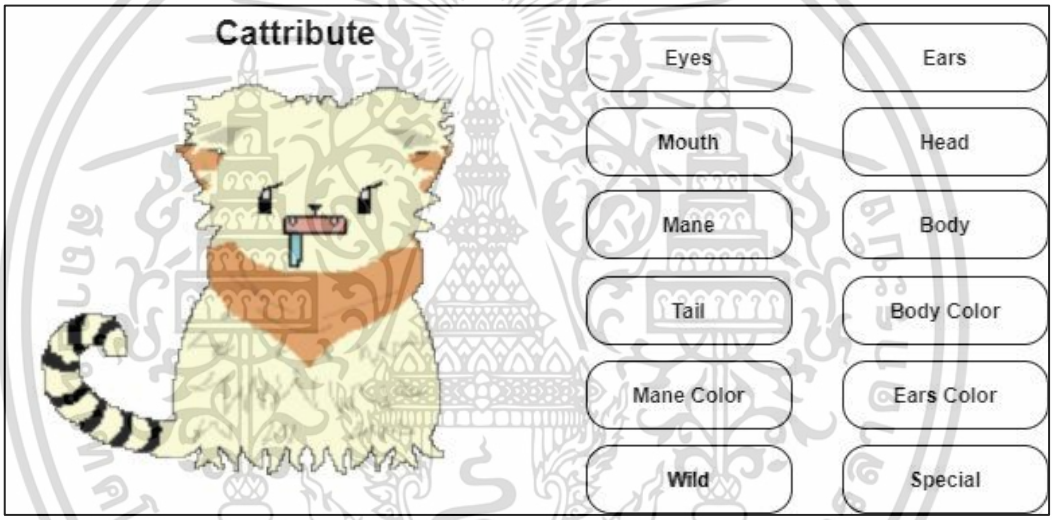
```
truffle(development)> let accounts = await web3.eth.getAccounts()
truffle console(development) > let receipt = await
kitty.createPromoKitty(accounts[0],“kitty”,0,0,0,0,””)
truffle console(development) > receipt
truffle console(development) > kitty.kitties(0)  //ตรวจสอบว่ามีเหรียญเกิดขึ้นจริง
```

4.3.3 การสร้างและทดสอบ GenesScience Contract สำหรับการเพาะพันธุ์แมวดิจิทัล

GeneScience คือ Contract ที่ช่วยเรื่องการผสมยีนของแมวดิจิทัล โดยที่จะไม่เปิดเผยให้คนทั่วไปรับรู้ เพื่อความสนุกในการเล่นแอปพลิเคชันนี้ โดยมีฟังก์ชัน mixGenes (uint256 _fatherGenes,uint256 _motherGenes) ที่รับ parameter ประเภท uint256 มา 2 ตัว คือ fatherGenes และ และ motherGenes โดยฟังก์ชันนี้จะนำยีนทั้งสองมาผสมกันโดยการแบ่ง uint256 ออกเป็น 48 ส่วนแต่ละส่วนมีความยาว 5 บิทได้ 240 บิท ที่เหลืออีก 16 บิทเป็นส่วนที่ไม่ต้องสนใจ และทั้ง 48 ส่วนนี้จะกำหนดลักษณะของแมวตัวนั้น 12 ลักษณะ ลักษณะละ 4 ส่วนโดยมี 1 ส่วน

เป็นยีนเด่น (genotype) ที่จะแสดงลักษณะออกมา อีก 3 ส่วนที่เหลือเป็นยีนด้อย(phenotype)ที่ไม่แสดงลักษณะแต่สามารถถ่ายทอดต่อไปยังลูกได้

ในฟังก์ชัน mixGene นี้มี while loop วน 48 รอบ เลือกยีนจากทั้งพ่อหรือแม่โดยที่มีค่าสุ่มค่าหนึ่งเป็นเลขจำนวนเต็มที่มีค่าตั้งแต่ 0 ถึง 99 เป็นตัวกำหนดในการเลือก ถ้าค่าสุ่มเป็นคู่ให้เลือกจากยีนแม่ เป็นเลขคี่เลือกจากยีนพ่อ จากนั้นสุ่มค่าอีกครั้งเพื่อเลือกยีนจาก 4 ยีนใน 1 ลักษณะมาถ่ายทอดสู่ลูกโดยมีลำดับความเป็นไปได้ที่ยีนลำดับแรก(ยีนเด่น) 50% และยีนต่อมา 25% , 12% , 12% ตามลำดับ และอีก 1% คือการกลายพันธ์(mutation) คือการทำให้ยีนนั้นแปลงไปเป็นยีนขั้นถัดไปได้ เมื่อวนครบ 48 ก็จะได้ยีนของลูกที่ได้รับการผสมพันธุ์เรียบร้อยแล้ว



รูป 4.25 ลักษณะทั้ง 12 ของแมวดิจิทัล

```
contract GeneScienceInterface{
    function mixGene(uint256 _fatherGenes,uint256 _motherGenes) public returns(uint256);
    function genesToArray(uint256 _genes) public pure returns(uint8[48] memory);
}
```

รูป 4.26 GenesScienceInterface สำหรับใช้ผสมพันธุ์

เมื่อสร้าง GeneScience Contract ขึ้นมาสำเร็จแล้ว ให้ KittyToken สร้าง instance ของ geneScience ขึ้นมาใน contract แล้วเรียกใช้ฟังก์ชันด้วยวิธีการ delegate call ระหว่าง contract ดัง

รูปที่ 4.22 ใน KittyToken Contract เรียกใช้ฟังก์ชัน mixGene ในฟังก์ชัน catMating เพื่อสร้างแมวตัวใหม่ขึ้นมา

```
function catMating(address _to,uint _fatherID,uint _motherID,string calldata _hash) external{
    require(_fatherID != _motherID, "can't breed with oneself");

    Kitty storage fatherKitty = kitties[_fatherID];
    Kitty storage motherKitty = kitties[_motherID];

    uint256 fatherGenes = fatherKitty.genes;
    uint256 motherGenes = motherKitty.genes;

    uint256 childGene = geneScience.mixGene(fatherGenes,motherGenes);

    uint256 childGeneration = fatherKitty.generation+1;
    if(fatherKitty.generation < motherKitty.generation){
      childGeneration = motherKitty.generation+1;
    }

    mintKitty(_to,"default Name",_fatherID,_motherID,childGeneration,childGene,_hash);
}
```

รูป 4.27 ฟังก์ชัน catMating สร้างแมวดิจิทัลขึ้นมาใหม่จากแมวตั้งต้น 2 ตัว

4.3.4 การสร้างและทดสอบ KittyMarket Contract สำหรับการซื้อขายแมวดิจิทัล

KittyMarket คือ Smart Contract เพื่อเป็นตลาดในการซื้อขายแมวดิจิทัล โดยให้เจ้าของแมวประกาศขายและกำหนดราคาลงในตลาด และผู้ที่ต้องการซื้อก็จะเข้ามาซื้อแมวตัวนั้น สิทธิความเป็นเจ้าของแมวก็จะถูกโอนจากผู้ขายไปยังผู้ซื้อ

เริ่มแรกสร้าง struct ชื่อ SalePost สำหรับเก็บค่าการประกาศขายแมว ดังรูปที่ 4.28

```
struct SalePost{
    uint256 weiPrice;
    _DateTime timeStamp;
    bool status;  //is on Sale or not
    uint256 kittyID;
}
```

รูป 4.28 ตัวแปร struct SalePost ประกาศขายแมวดิจิทัล

เงื่อนไขของการขายแมวคือ ผู้ขายต้องเป็นเจ้าของแมวดิจิทัลตัวนั้น และแมวต้องไม่ถูกประกาศขายมาก่อน(ค่า isOnSale ใน KittyToken แสดงค่าว่าประกาศขายอยู่หรือไม่) เมื่อเงื่อนไขครบจะสร้าง SalePost ขึ้นมาเก็บไว้ในอาเรย์ SalePost ซึ่งก่อนที่ผู้ขายจะประกาศขายได้ ผู้ขายต้องให้ approval กับ contract เพื่อให้ KittyMarket Contract มีสิทธ์ในการโอนแมวดิจิทัลของผู้ขายไปให้ผู้ซื้อ ด้วยการเรียกใช้ฟังก์ชัน approve(address to,uint256 tokenID) ไปยัง KittyToken Contract ก่อน

เมื่อผู้ซื้อมาซื้อจะเรียกใช้ฟังก์ชัน buyKitty มีเงื่อนไขคือ 1.ผู้ประกาศขายไม่สามารถซื้อของที่ตัวเองประกาศขายได้ 2. ผู้ใช้ต้องสร้าง transaction ที่กำหนดค่าของ ETH Value ที่ส่งมา ต้องเท่ากับราคาที่ประกาศไว้ แล้วเมื่อเรียกใช้ฟังก์ชันสำเร็จ ETH จะถูกโอนไปให้ผู้ขาย และเรียกใช้ฟังก์ชันภายใน completeSale(uint256 SaleID) ซึ่งทำหน้าที่ปิดประกาศขายและโอนแมวดิจิทัลไปให้ผู้ซื้อ และเนื่องจากข้อจำกัดของฟังก์ชัน transferFrom ใน interface ERC721 ซึ่งผู้ใช้งานต้องเป็นเจ้าของหรือผู้ที่ได้รับอนุญาตของเหรียญหรือในที่นี้คือแมวที่ต้องการโอนเท่านั้น ทางผู้พัฒนาจึงปรับปรุง KittyContract โดยสร้างตัวแปร address ที่ชื่อ superAuthority และฟังก์ชันชื่อ setMarketContract และดัดแปลงฟังก์ชัน transferFrom ให้ marketContract สามารถโอนแมวดิจิทัลไปให้ผู้ซื้อได้

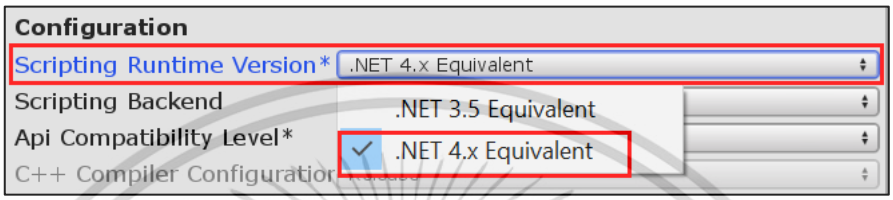
4.4 การสร้างและทดลอง Mobile Application ด้วย Unity

4.4.1 การนำเข้าไลบรารี่ชื่อ Nethereum ลงใน Unity Project

Nethereum คือไลบรารี่ของ .Net Framework ที่ช่วยแอปพลิเคชัน .Net สามารถติดต่อกับเครือข่ายอีเธอเรียมได้ ซึ่งใน Unity ใช้ C# .Net เป็นภาษา Scripting ที่เชื่อมต่อกับ GameEngine ของ Unity แต่วิธีการนำเข้าของ Unity ต่างจากแอปพลิเคชันของ .Net ทั่วไปที่สามารถนำเข้าผ่าน Package Manager ชื่อว่า Nuget เพราะการนำเข้าไลบรารี่ของ Unity จำเป็นต้องนำเข้าไฟล์ประเภท .dll โดยตรงและต้อง สร้างไฟล์ชื่อ link.xml ขึ้นบนโฟลเดอร์ asset ใน Unity Project ดังรูปที่ 4.29 และจำเป็นต้องเปลี่ยน Scripting Runtime Version ใน Unity Project ให้เป็น .Net 4.0 ดังรูปที่ 4.30

```
link.xml
1  <linker>
2    <assembly fullname="System.Core">
3      <type fullname="System.Linq.Expressions.Interpreter.LightLambda" preserve="all" />
4    </assembly>
5  </linker>
```

รูป 4.29 linker.xml



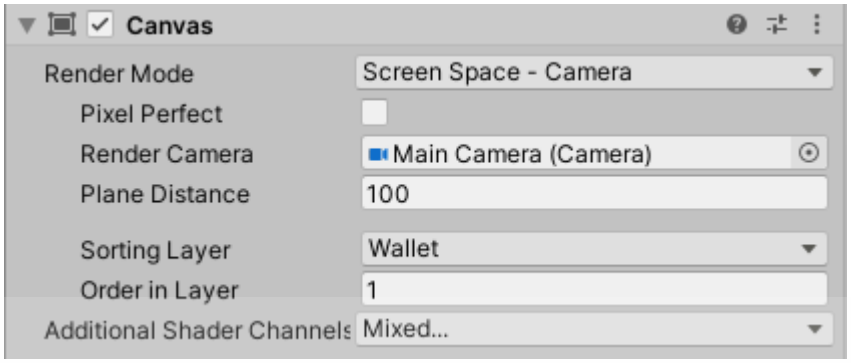
รูป 4.30 เปลี่ยน Scripting Runtime Version ใน Unity Player Setting

เพื่อให้ง่ายต่อการนำไฟล์ .dll ของไลบรารี่ที่ต้องการทั้งหมด จึงสร้าง .Net Project ชั่วคราวขึ้นมาด้วย Visual Studio และติดตั้ง Nethereum ผ่าน Package Manager จะได้ไฟล์ .dll ที่ต้องการทั้งหมดโดยที่ไม่ต้องดาวน์โหลดทีละไฟล์ จากนั้นจึงคัดลอกไฟล์เหล่านั้นไปยังโฟลเดอร์ Asset/Plugins ใน Unity Project เป็นอันเรียบร้อย

4.4.2 การสร้างหน้าเกมหลัก

หน้าเกมคือหน้าที่นำแมวดิจิทัลมาเลี้ยง โดยที่ผู้เล่นสามารถ ให้อาหาร อาบน้ำ ทำความสะอาดห้อง และเล่นกับมัน ในหน้านี้จะมีปุ่มกดเพื่อใช้ฟังก์ชันเลี้ยงแมวนี้อยู่ด้านล่าง 4 ปุ่ม ด้านบนจะมีหัวใจ 5ดวง แสดงคะแนนค่าความรักของแมวตัวปัจจุบันที่มีต่อเรา เมื่อกดปุ่มฟังก์ชันเหล่านั้นผู้เล่นจะได้คะแนนความรัก และคะแนนความรักนี้จะนำไปใช้เป็นเงื่อนไขในการใช้ฟังก์ชันของ smart contract เช่น ต้องมีหัวใจครบ 3 ถึงจะประกาศขายได้ หรือ ต้องมีหัวใจครบ 5 ดวงถึงนำมาผสมพันธุ์ได้

เริ่มแรกในการสร้าง UI บน Unity จำเป็นต้องสร้าง GameObject ที่ชื่อว่า Canvas และในหน้าด้านขวา ปรับ Render Mode ใน Canvas Component เป็น Screen Space – Camera เพื่อให้ Canvas เรนเดอร์ UI ออกมาตาม Main Camera ดังรูปที่ 4.31



รูป 4.31 Canvas Component

จากนั้นสร้าง UI object ตามที่ออกแบบไว้ คือ 1. Action Button 4 อันด้านล่าง 2. HeartGroup เป็น Image สำหรับรูปหัวใจที่เป็นคะแนน 3. Wallet Button ปุ่มที่ใช้เรียกหน้า Wallet ขึ้นมา 4. Cat Button ปุ่มสำหรับเปิดรายการของแมวที่ผู้เล่นมีอยู่ จึงได้ผลออกมาดังในรูปที่ 4.32

รูป 4.32 สร้าง UI Object ทั้งหมดขึ้นมา

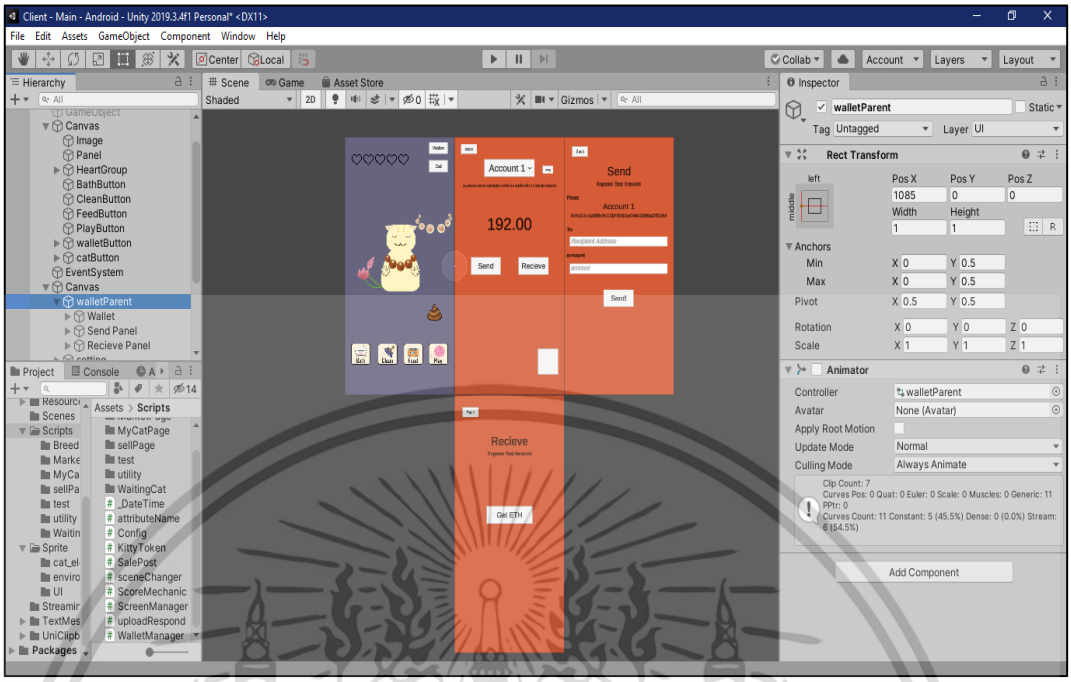
ต่อมาสร้าง C# script ที่ใช้ในการเล่นเกมหลักอย่าง อาบน้ำ ทำความสะอาด ให้อาหาร และเล่น ปุ่มแต่ละปุ่มเมื่อกดไปแล้วจะเข้าสู่สถานะ disable หรือการไม่ได้ชั่วคราว และทำการนับเวลาถอยหลังเพื่อกลับมาใช้ได้อีกรอบ วิธีการคือ เมื่อกดปุ่มแล้วจะบันทึกเวลา UNIXTimeStamp ลงใน

ตัวแปล Global Key-Value ของ Unity ที่ชื่อ PlayerPrefs แล้วเช็คในทุก ๆ เฟรมว่าหมดเวลาแล้วหรือไม่ ฉะนั้นเมื่อผู้เล่นออกจากแอปพลิเคชันแล้วกลับเข้ามาใหม่เวลาก็ยังคงนับตามจริง

แมวที่อยู่ตรงกลางของหน้าเป็น GameObject ที่มี Component ชื่อ Sprite Renderer ซึ่งแมวตัวนี้เกิดจากนำ sprite หลายองค์ประกอบมาต่อกัน แมวตัวนี้คือแมวที่ผู้เล่นเลือกมาเล่นในฟังก์ชันหลัก และเมื่อผู้เล่นเลือกแมวจากหน้า Catalog มี GameObject ชื่อ gameManager จะบันทึกค่ายีนของแมวตัวที่เลือกไว้ใน PlayerPrefs ด้วย key ชื่อ selectedKittyID

4.4.3 การสร้างหน้า Wallet ทดลองเชื่อมต่อกับ Smart Contract

หน้า wallet คือที่ให้ผู้เล่นได้ใช้ฟังก์ชัน wallet ของอีเธอเรียมได้อย่างเต็มรูปแบบ wallet ผู้เล่นสามารถโอน รับ สร้างบัญชี เปลี่ยนบัญชี และใช้ในการจัดการ transaction ที่ใช้ติดต่อกับ smart contract สำหรับ UI หน้านี้แบ่งออกเป็น 3 ส่วนคือ 1. หน้าสำหรับเช็คยอกเงินและเปลี่ยนบัญชี ดูเลข address 2. หน้าสำหรับโอน ETH ไปยัง address อื่น 3. หน้าสำหรับ รับ ETH จากผู้อื่น ในที่นี้จะรับจากบริการแจกจ่าย ETH บนเครือข่ายอีเธอเรียมทดลอง (ETH faucet) โดยที่หน้า UI ทั้ง 3 นี้อยู่ใน scene เดียวกับหน้าหลักแต่คนละ Canvas เพื่อที่จะไปต้องเปลี่ยน scene และฟังก์ชันในหน้าหลักยังคงดำเนินไปตามปกติ และ Canvas ของหน้า wallet จะถูก render ทับหน้าหลัก ดังรูปที่ 4.33



รูป 4.33 โครงสร้างของหน้า wallet อยู่ scene เดียวกับหน้าหลัก

ในหน้า wallet นี้จะมีการติดต่อกับเครือข่ายอีเธอเรียมผ่านไลบรารี่ Nethereum เริ่มแรกโดยการสร้าง GameObject ชื่อว่า WalletManager ขึ้นมาและ C# script Component ขึ้นมาและใน script นั้นมี instance ของ Nethereum.Web3 ซึ่งเป็น object ที่ใช้เพื่อการติดต่อกับเครือข่ายอีเธอเรียม จากนั้นสร้าง instance ของ wallet ซึ่งใช้ในการจัดการกับคู่กุญแจ (key pair) โดยรับ parameter เป็น string ของ seed phrase (กลุ่มคำ 12 คำที่ใช้ในการสร้าง HDWallet) จากนั้นสามารถสร้างฟังก์ชันรับค่า balance มาแสดงใน wallet หน้าแรกได้ดังรูปที่ 4.34 เป็นฟังก์ชัน updateBalance() ทำหน้าที่อ่านค่า ETH balance ของบัญชีปัจจุบันที่ผู้เล่นสามารถเลือกได้จาก dropdown ที่อยู่ตรงกลางของหน้า จากเครือข่ายอีเธอเรียมมาในหน่อยของ wei (10^18 wei เท่ากับ ether) จากนั้นแปลงจากหน่วย wei เป็นหน่วย ether ด้วย

```
1 reference
public async void updateBalance(){
    var balance = await web3.Eth.GetBalance.SendRequestAsync(currentAccount.Address);
    var etherAmount = Web3.Convert.FromWei(balance.Value);
    balanceText.SetText($"{etherAmount.ToString("0.00")} ETH");
}
```

รูป 4.34 ฟังก์ชัน updateBalance

ต่อมาในหน้า sendEther ที่ผู้เล่นสามารถส่ง ETH ไปให้บัญชีอื่นได้ โดยที่การส่ง ETH นี้ทำให้เกิดการเปลี่ยนแปลงข้อมูลในเครือข่ายอีเธอเรียม ต่างจาก updateBalance ที่อ่านค่าจากเครือข่ายมาอย่างเดียว จึงต้องสร้าง transaction ส่งไปยังเครือข่าย โดยใช้ฟังก์ชัน Nethereum.Web.Eth.GetEtherTransferService().TransferEtherAndWaitForRecieptAsync() รับ parameter 2 ตัว คือ 1. toAddress บัญชีปลายทางที่รับ ETH 2. Amount จำนวน ETH ในหน่าย wei ดังรูปที่ 4.35

```
1 reference
public async void sendEther(decimal amount){
    var web3 = new Web3(currentAccount,config.web3Provider);
    Debug.Log("transaction pending");
    var transaction = await web3.Eth.GetEtherTransferService()
                        .TransferEtherAndWaitForReceiptAsync(
                            toAddressField.text,
                            amount
                        );
    Debug.Log("done");
    Debug.Log(transaction.TransactionHash);
}
```

รูป 4.35 ฟังก์ชัน sendEther

หน้าสุดท้ายคือหน้า recieveEther เนื่องแอปพลิเคชันนี้ deploy อยู่บนเครือข่ายทดลองชื่อ ropsten ผู้เล่นสามารถขอ ETH ฟรีจำนวน 1 ETH ต่อวัน ได้จากเว็บไซต์ https://faucet.ropsten.be/ และในหน้านี้มีปุ่ม Get ETH ซึ่งใช้ฟังก์ชัน getEther ใน WalletManager ดังรูปที่ 4.36 ทำหน้าที่ส่ง HTTP GET request ไปยัง

```
0 references
IEnumerator GetEther() {
    UnityWebRequest www = UnityWebRequest.Get($"http://localhost:3000/request_ether/{currentAccount.Address}");
    yield return www.SendWebRequest();

    if(www.isNetworkError || www.isHttpError) {
        Debug.Log(www.error);
    }
    else {
        // Show results as text
        Debug.Log(www.downloadHandler.text);
        getEtherButton.interactable = true;
        waitMomentText.SetActive(false);

        // Or retrieve results as binary data
        byte[] results = www.downloadHandler.data;
    }
}
```

รูป 4.36 ฟังก์ชัน getEther

4.4.4 การสร้างหน้า myCat

หน้า myCat คือหน้าสำหรับแสดงรายการแมวที่ผู้เล่นมีอยู่ โดยอ่านค่าจาก KittyToken contract ซึ่งใน contract ไม่ได้เก็บรายการของบัญชีใดบัญชีหนึ่งไว้ แต่มีแค่เพียงอาเรย์ Kitty[] public kitties และ mapping (uint256 => address) internal _tokenOwner ซึ่งเป็น key-value array ที่แมพไอดีของแมวดิจิทัลไปยังบัญชีของเจ้าของ เนื่องจาก smart contract จำเป็นต้องเรียบง่ายที่สุด จึงไม่เก็บอาเรย์ของแมวที่ครอบครองของเจ้าของนั้นไว้ และไม่มีฟังก์ชันที่ใช้หาแมวที่เจ้าของครอบครอง จึงต้องสร้างฟังก์ชันหารายการของแมวที่เจ้าครอบครองในฝั่งผู้ใช้งานแทน

เริ่มแรกสร้างฟังก์ชันที่ใช้อ่านค่าข้อมูลของแมวดิจิทัล 1 ตัว ชื่อ GetKitty(index) รับ parameter 1 ตัว คือ index ของแมวดิจิทัลในอาเรย์ kitties ดังรูปที่ 4.37

```
public async Task<KittyToken> GetKitty(long index)
{
    var kittyFunction = KittyTokenContract.GetFunction("kitties");
    var kitty = await kittyFunction.CallDeserializingToObjectAsync<KittyToken>(index)
                                   .ConfigureAwait(false);
    return kitty;
}
```

รูป 4.37 ฟังก์ชัน GetKitty

จากนั้นสร้างฟังก์ชันที่อ่านค่าจาก smart contract อีก 2 ค่า คือ 1. balanceOf(address) จำนวนแมวดิจิทัลที่ครอบครองอยู่ของของบัญชีนั้น 2. tokenOfOwnerByIndex(address,index) อ่านค่า ID

ของแมวดิจิทัล จาก index ของรายการของแมวที่บัญชีนั้นครอบครองอยู่ จากนั้นก็สามารถสร้างฟังก์ชันที่อ่านค่ารายการของแมวที่ครอบครองอยู่ออกมาได้ ดังรูปที่ 4.38

```
public async Task<List<KittyToken>> GetAllOwnedKitty(string address)
{
    var balance = await GetBalanceOf(address).ConfigureAwait(false);

    var kitties = new List<KittyToken>();

    for (long i = 0; i < balance; i++)
    {
        long kittyIndex = await GetTokenOfOwnerByIndex(address,i);
        kitties.Add(await GetKitty(kittyIndex).ConfigureAwait(false));
    }
    return kitties;
}
```

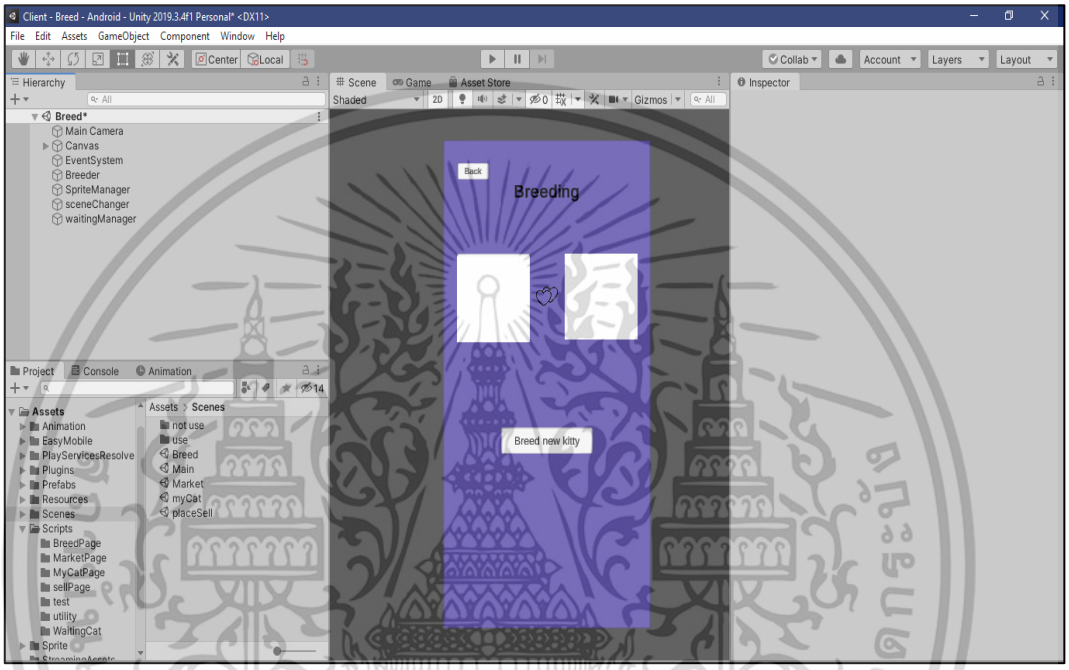
รูป 4.38 ฟังก์ชัน GetAllOwnedKitty

เมื่อสามารถอ่านค่าแมวดิจิทัลที่ผู้เล่นครอบครองได้ทั้งหมดแล้ว จึงนำมาแสดงในฝั่งผู้ใช้งาน เริ่มแรกสร้าง Scroll View ขึ้นมาบน Unity เพื่อใช้สำหรับเป็น container แสดงรายการแมวดิจิทัล จากนั้นสร้าง object ที่แสดงข้อมูลของแมวแบบย่อแล้วบันทึกไว้เป็น prefab ใน project ดังรูปที่ 4.39 ต่อมาเขียนฟังก์ชันชื่อ generateCatList() ในไฟล์ชื่อ catListManager.cs สร้างรายการของแมว โดยใช้ข้อมูลจากฟังก์ชัน GetAllOwnedKitty มาแสดงบน Scroll View ที่ขึ้นมา

รูป 4.39 GameObject แสดงรายละเอียดของแมวดิจิทัลแบบย่อ

4.4.5 การสร้างหน้า Breed

หน้า Breed คือหน้าที่ใช้สำหรับเพาะพันธุ์แมวดิจิทัล ในหน้านี้มีปุ่มหลัก 2 ปุ่มเพื่อให้ผู้เล่นเลือกแมวที่จะมาผสมพันธุ์ และอีก 1 ปุ่ม เพื่อเริ่มการผสมพันธุ์แมว เริ่มสร้างหน้านี้ขึ้นมาดัง รูปที่ 4.40



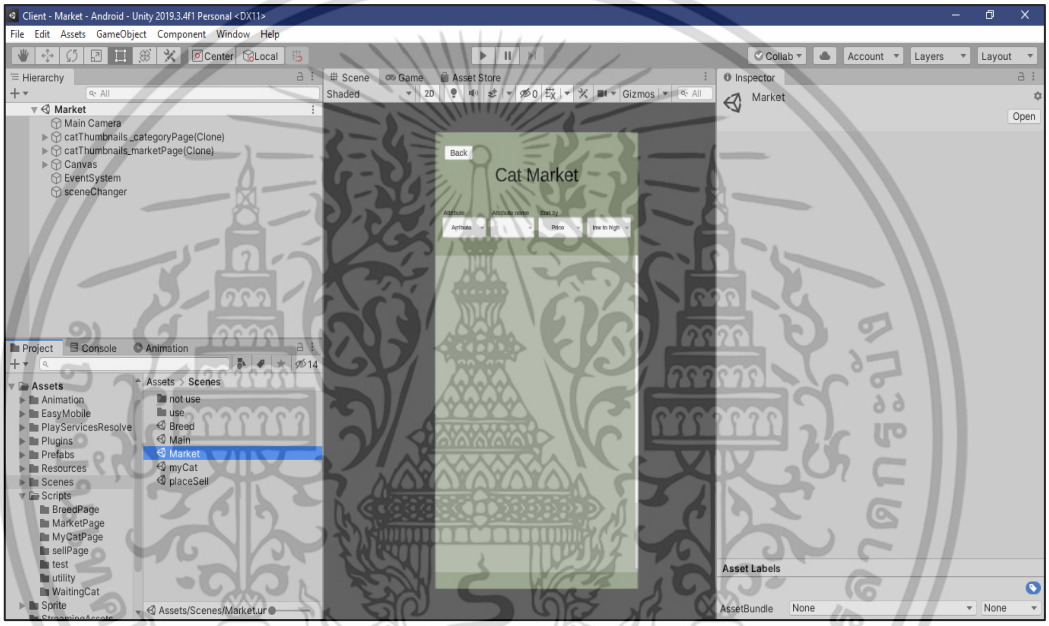
รูป 4.40 เริ่มต้นสร้างหน้า Breed

จากนั้นสร้าง GameObject ชื่อ selectManager เพื่อควบคุมการเลือกแมวที่นำมาผสมพันธุ์ และนำ catScrollView จากหน้า myCat มาเปลี่ยนแปลงและนำมาใช้ โดยที่เมื่อกดที่รายการของแมว selectManager จะเลือกตัวนั้นไว้ และปิด catScrollView ไปแล้วแสดงรูปของแมวตัวที่เลือกมาบนปุ่มทั้งสอง จากนั้นสร้าง GameObject ชื่อ Breeder และ C# script ชื่อ catBreeder ทำหน้าที่ในการติดต่อกับ KittyToken contract เพื่อเรียกใช้งาน catMating เพื่อผสมพันธุ์แมวทั้งสองตัวที่เลือกมาและสร้างลูกออกมาใหม่ เมื่อกดปุ่ม Breed new kitty แล้ว catBreeder จะส่ง transaction เพื่อเรียกใช้ catMating ไปยังเครือข่าย และเปลี่ยนไปยังหน้ารอการตรวจสอบ transaction และเมื่อ transaction ได้รับการตรวจสอบ Breeder จะทำการสร้างรูปของแมวตัวที่เกิดใหม่ ด้วยการนำ sprite ชิ้นส่วนของแมวดิจิทัล เช่น ตา หู ปาก หัว ตัว แผงคอ หาง มาประกอบกันแล้วใช้วิธีการ บันทึกภาพหน้าจอ

(Screen Capturing) ในการบันทึกภาพของแมวที่นำมาต่อกัน จากนั้นก็จะส่งภาพนี้ไปยังเครือข่าย IPFS

4.4.6 การสร้างหน้า Market

หน้า Market คือหน้าที่แสดงรายการประกาศขายแมวดิจิทัล ในหน้านี้ประกอบด้วย Dropdown 4 อันและ Scroll View ที่แสดงรายการประกาศขาย เริ่มแรกด้วยสร้างหน้าให้มีองค์ประกอบตามที่ออกแบบ ดังรูปที่ 4.41 ขึ้นมาบน Unity



รูป 4.41 เริ่มสร้างหน้า market

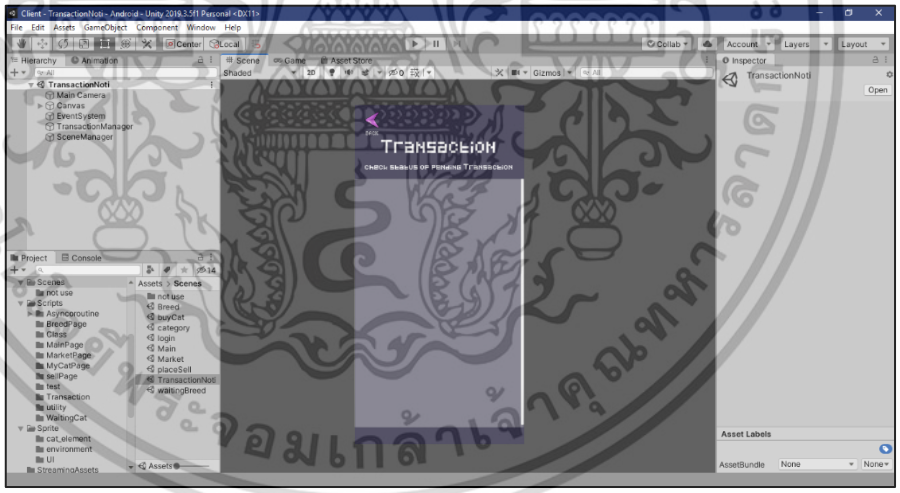
จากนั้นนำ catScrollView ในหน้า myCat มาเปลี่ยนแปลงและนำมาใช้ ด้วยการอ่านค่าประกาศขายจาก KittyMarket contract โดยที่ใน contract นี้มีตัวแปรชื่อ salePost ดังรูปที่ 4.28 โดยเริ่มแรกเมื่อผู้เล่นเข้ามาในหน้านี้ catScrollView จะแสดง salePost 10 อันล่าสุดใน KittyMarket จากนั้นเมื่อผู้เล่นกดปุ่ม filter และเลือกตัวเลือกเสร็จ catScrollView จะอัพเดทตามตัวกรอง (filter) นั้นตามฟังก์ชัน updateSalePostList ในสคริปต์ที่ชื่อ MarketManager.cs ที่ใช้จัดการหน้า Market นี้ใน Unity และตัวกรอง 4 ปุ่มนั้นประกอบไปด้วย 1. Attribute เลือกลักษณะของแมวดิจิทัลที่มีอยู่ 12 ลักษณะที่ต้องการให้กรอง 2. Attribute Name เลือกชื่อของลักษณะในข้อที่ 1 เพื่อตามการกรอง เมื่อตัวกรองในข้อที่ 1 และ 2 เสร็จแล้วจะทำการอัพเดท catScrollView ตามตัวกรองนั้น 3. SortBy

เลือกตัวแปรที่ต้องการให้เรียงตาม เช่น ราคา , ID , วันเกิด 4. เลือกรูปแบบการเรียงในข้อที่ 3 จากน้อยไปมากหรือมากไปน้อย

4.4.7 การสร้างหน้า Transaction Notification

เนื่องเครือข่ายบล็อกเชนอย่างเครือข่ายอีเธอเรียมเป็นระบบที่มีความปลอดภัยสูง จึงเกิดความล่าช้าที่ในการยืนยัน Transaction มากกว่าระบบที่มีศูนย์กลางเมื่อส่ง Transaction ไปยังเครือข่ายแล้วผู้ใช้งานต้องรอมากกว่า 20 วินาทีในการยืนยัน transaction นั้นซึ่งส่งผลเสียต่อประสบการณ์ในการใช้งาน ดังนั้นเมื่อส่ง Transaction ไปยังเครือข่ายแล้วให้เปลี่ยนไปยังหน้าหลัก จากนั้นแจ้งผู้ใช้งานในภายหลังเมื่อ Transaction ได้รับการยืนยันแล้ว

เริ่มด้วยสร้าง Scene ชื่อว่า TransactionNotification ใน Scene นี้มี ScrollView ที่แสดงรายละเอียดโดยชื่อย่อของ Transaction ที่ผู้ใช้งานได้สร้างขึ้นในอดีต โดยในแต่ละ transaction แสดง 1. ชื่อของฟังก์ชันที่ใช้งาน 2. แมวที่เกี่ยวข้องกับฟังก์ชันนั้น 1 หรือ 2 ตัว 3. สถานะของ Transaction มี Success , Fail หรือ Pending ดังรูปที่ 4.42



รูป 4.42 หน้า TransactionNotification

ข้อมูลของ Transaction ที่ผู้เล่นสร้างขึ้นมาจะถูกเก็บไว้ในไฟล์ PendingTransaction.json ไฟล์นี้เก็บอาร์เรย์ของ TransactionData class ดังรูปที่ 4.43 ใน TransactionData มีค่า hash หรือ transactionHash ในเครือข่ายอีเธอเรียม เมื่อกดที่ลิสต์ของ Transaction ในรูปที่ 4.42 แอปพลิเคชันจะ

เปิดลิงก์บนบราวเซอร์ https://ropsten.etherscan.io/tx/<transactionHash> เพื่อให้ผู้ใช้เช็คสถานะของ Transaction นั้น

```
[Serializable]
8 references
public class TransactionData{

    4 references
    public TransactionData(bool _status,string _hash,string _function,string _catID1,string _catID2){
        WatchedStatus = _status;
        hash =_hash;
        function = _function;
        catID1 = _catID1;
        catID2 = _catID2;
    }
    3 references
    public bool WatchedStatus;
    5 references
    public string hash;
    2 references
    public string function;
    2 references
    public string catID1;
    2 references
    public string catID2;
}
```

รูป 4.43 TransactionData Class

4.5 การทดลองและผลลัพธ์

เนื่องจากการพัฒนาแอปพลิเคชันสำหรับการเรียนรู้และใช้งานจริง จำเป็นต้องมีการทดสอบแอปพลิเคชันเพื่อนำผลการทดลองไปแก้ไขและปรับปรุงงานต่อไป

4.5.1 วิธีการทดลอง

ให้ผู้ทดสอบทั้งหมด 5 คนได้ทำแบบทดสอบความเข้าใจเรื่องเทคโนโลยีบล็อกเชนชุดเดียวกัน ซึ่งเป็นคำถามให้เลือกตอบทั้งหมด 10 คำถาม ก่อนจะให้ผู้ทดสอบทดลองใช้งานแอปลิเคชันและหลังจากนั้นจะให้ทำแบบทดสอบชุดเดิมเพื่อประเมินผล

4.5.2 แบบทดสอบสำหรับการประเมินความเข้าใจเทคโนโลยีบล็อกเชน

ส่วนคำถามเลือกตอบได้คัดเลือก 10 คำถามมีดังนี้

1. ทำไมผู้เล่นเปลี่ยนลักษณะของแมวที่ครอบครองไม่ได้

1) เพราะลักษณะแมวที่จะเปลี่ยนนั้นซ้ำกับของผู้เล่นอื่น

2) เพราะข้อมูลในบล็อกเชนไม่สามารถเปลี่ยนแปลงได้

3) เพราะต้องมีไอเทมในเกมสำหรับเปลี่ยนลักษณะของแมวก่อน

2. ผู้เล่นสามารถทำการโอนเงินหน่วย ETH ให้กับผู้เล่นอื่นได้โดยตรงเลยหรือไม่

1) ได้

2) ไม่ได้

3. ทำไมในเกมนี้ผู้เล่นสามารถโอนแมวกับผู้อื่นได้อย่างอิสระ

1) เพราะบล็อกเชนเป็นระบบแบบกระจายศูนย์ ซึ่งสามารถสร้างข้อมูลใหม่มาต่อในระบบได้อย่างอิสระ

2) เพราะข้อมูลแมวถูกทำสำเนาได้

3) เพราะอินเตอร์เน็ตเข้าถึงกันได้หมด

4. รหัสในการใช้งาน Wallet เป็นรูปแบบใด

1) Password ที่ผู้เล่นตั้งขึ้นมาเอง

2) กลุ่มตัวอักษรภาษาอังกฤษ 12 ตัว

3) ได้รับรหัสกู้คืนจากระบบผ่านอีเมล

4) คำถามสำรองและคำตอบที่ผู้เล่นตั้งค่าไว้

5. ขั้นตอนการซื้อขายแมวต้องทำอย่างไร

1) ตกลงกับผู้ซื้อ-ขาย และทำการซื้อขาย ณ เวลานั้น

2) ผู้ขายตั้งประกาศขายทิ้งไว้ แล้วผู้ซื้อมาซื้อตามที่ผู้ขายประกาศไว้

3) ผู้ซื้อตั้งประกาศความต้องการซื้อไว้แล้วผู้ขายจึงเข้ามาขาย

4) ผู้ซื้อโอนเงินให้ผู้ขายแล้วผู้ขายโอนแมวให้ ณ เวลาที่ทำการซื้อขาย

6. ถ้าผู้เล่นทำรหัสในการกู้คืน Wallet สูญหายควรทำอย่างไร

1) ขอรหัสใหม่ผ่านอีเมล

2) ตอบคำถามสำรองตามที่ผู้เล่นตั้งค่าไว้

3) ไม่สามารถกู้คืนมาได้หากลืมรหัสผ่าน

7. ท่านจะทราบข้อมูลอะไรในหน้าต่าง Transaction Notification

1) การแจ้งเตือนเหตุการณ์ที่เกิดขึ้นของแมว เช่น แมวหิว

2) แสดงรายการธุรกรรมที่ทำไว้ ชื่อของฟังก์ชันที่ใช้งาน และสถานะการยืนยัน

3) แสดงหมายเลขของธุรกรรมที่ทำไปแล้วอย่างเดียว

4) แสดงลักษณะของแมวที่ครอบครอง

8. ในเวลาที่ท่านทำธุรกรรม (Transaction) เช่น ซื้อ ขาย เพาะพันธุ์ ธุรกรรมนั้นได้รับการยืนยันทันทีเลยหรือไม่

1) ใช่

2) ไม่ใช่

9. ท่านลบแมวที่มีออกไปจากระบบได้หรือไม่ เพราะอะไร

1) ได้ เพราะข้อมูลดิจิทัลถูกลบได้เสมอ

2) ได้ เพราะระบบสร้างไว้ให้สามารถลบได้

3) ไม่ได้ เพราะระบบของเทคโนโลยีบล็อกเชนไม่สามารถลบข้อมูลได้

4) ไม่ได้ เพราะระบบไม่ได้สร้างให้ลบ

10. ท่านสามารถสร้าง Account ใหม่ใน Wallet เดิมได้หรือไม่

1) สร้างได้

2) สร้างไม่ได้

ส่วนประเมินให้คะแนนความเข้าใจในเทคโนโลยีบล็อกเชนหลังทดลองแอปพลิชันมี 2 ข้อดังนี้

1. ท่านให้คะแนนความเข้าใจในเทคโนโลยีบล็อกเชนเท่าใด (1-10 คะแนน)

2. ท่านอยากใช้งานเทคโนโลยีบล็อกเชนมากเท่าใด (1-10 คะแนน)

4.5.3 ผลการทดลอง

จากการผู้ทดสอบ 5 คนมีผลลัพธ์ออกมาดังนี้

1. สุวิดา จันทนานนท์ อายุ 31 ปี

คะแนนทดสอบ

ก่อนทดลองแอปพลิเคชัน 3 คะแนน

หลังทดลองแอปพลิเคชัน 6 คะแนน

ความมั่นใจในความรู้เทคโนโลยีบล็อกเชน 5 คะแนน

ความอยากใช้งานเทคโนโลยีบล็อกเชน 6 คะแนน

ความคิดเห็นเพิ่มเติม : รูปประกอบเกมน่ารัก แถมมีความรู้แทรก แต่ตัวหนังสือเยอะมากเกินไปควรลดให้น้อยลง

2. จินตนา จันทนานนท์ อายุ 61 ปี

คะแนนทดสอบ

ก่อนทดลองแอปพลิเคชัน 2 คะแนน

หลังทดลองแอปพลิเคชัน 5 คะแนน

ความมั่นใจในความรู้เทคโนโลยีบล็อกเชน 1 คะแนน

ความอยากใช้งานเทคโนโลยีบล็อกเชนความอยากใช้งาน 3 คะแนน

ความคิดเห็นเพิ่มเติม : ไม่ค่อยเก่งเรื่องเทคโนโลยี เข้าใจยาก แต่เกมดูน่ารักดีเพราะส่วนตัวชอบแมว

3. ณัฐวัลย์ สายวรรณ์ อายุ 24 ปี

ก่อนทดลองแอปพลิเคชัน 4 คะแนน

หลังทดลองแอปพลิเคชัน 5 คะแนน

ความมั่นใจในความรู้เทคโนโลยีบล็อกเชน 3 คะแนน

ความอยากใช้งานเทคโนโลยีบล็อกเชนความอยากใช้งาน 4 คะแนน

ความคิดเห็นเพิ่มเติม : บล็อกเชนเป็นเทคโนโลยีที่น่าสนใจ แต่คิดว่ายังเข้าใจยาก เนื่องจากไม่เคยรู้จักบล็อกเชนมาก่อน ควรแก้หน้าตาเกมให้ดูดีกว่านี้

4. ณัฐนันท์ อริยะเมตตา อายุ 21 ปี

คะแนนทดสอบ

ก่อนทดลองแอปพลิเคชัน 5 คะแนน

หลังทดลองแอปพลิเคชัน 8 คะแนน

ความมั่นใจในความรู้เทคโนโลยีบล็อกเชน 6 คะแนน

ความอยากใช้งานเทคโนโลยีบล็อกเชนความอยากใช้งาน 5 คะแนน

ความคิดเห็นเพิ่มเติม : ภาพรวมออกมาสวยดี ได้ความรู้เพิ่มเติม แต่Filter การซื้อแมวยังดูยาก อยากให้แก้ไขหน้า UI และลดตัวหนังสือใน Tutorial ลง

5. ฐาปนีย์ บุญชอบ อายุ 23 ปี

คะแนนทดสอบ

ก่อนทดลองแอปพลิเคชัน 6 คะแนน

หลังทดลองแอปพลิเคชัน 7 คะแนน

ความมั่นใจในความรู้เทคโนโลยีบล็อกเชน 5 คะแนน

ความอยากใช้งานเทคโนโลยีบล็อกเชนความอยากใช้งาน 5 คะแนน

ความคิดเห็นเพิ่มเติม : ชอบที่แมวมีความหลากหลาย แต่ตัวอักษรเยอะไป ไม่ชอบอ่านเยอะ อยากให้อนาคตเพิ่ม UX ให้สวยงามยิ่งขึ้นและเพิ่มแมว Special ให้มากยิ่งขึ้น

บทที่ 5
สรุปผลและปัญหาที่พบ

5.1 สรุปผลการทดลองและสรุปผลการทำงาน

จากการทดลองกับผู้ใช้งานจริงมีผลลัพธ์ในภาพรวมคือการได้ทำความรู้จักเทคโนโลยีบล็อกเชนภายในตัวเกม แต่เนื่องจากเทคโนโลยีนี้ยังไม่ได้ถูกใช้งานในชีวิตประจำของผู้ทดลอง จึงถือเป็นเรื่องที่ต้องใช้เวลาในการเรียนรู้ หากพัฒนาต่อไปคาดว่าเกมนี้จะช่วยลดเวลาในการทำความเข้าใจบล็อกเชนยิ่งขึ้น

สรุปการทำงานของแอปพลิเคชันนี้แบ่งออกเป็น 2 ส่วน คือส่วนของ Smart Contract ที่อยู่บนเครือข่ายอีเธอเรียม และส่วนติดต่อกับผู้ใช้งานที่เป็นแอปพลิเคชันมือถือระบบแอนดรอยด์

5.1.1 ส่วนของ Smart Contract ที่อยู่บนเครือข่ายอีเธอเรียม

ส่วนนี้ทำหน้าที่เป็นฐานข้อมูลสาธารณะ เพื่อเก็บข้อมูลของสินทรัพย์ดิจิทัล ซึ่งในที่นี้คือแมวดิจิทัลที่มีลักษณะต่างกัน กำหนดโดยค่ายีน บันทึกอยู่ใน Smart Contract ที่ชื่อว่า KittyToken บนเครือข่ายอีเธอเรียมทดลองที่ชื่อ Ropsten ใน KittyToken นอกจากทำหน้าที่เก็บข้อมูลของแมวดิจิทัลแล้วยังสามารถสร้างแมวดิจิทัลขึ้นมาใหม่ ด้วยฟังก์ชันผสมพันธุ์ (Breeding) ด้วยการนำยีนของแมวดิจิทัล 2 ตัว มาผสมกันและสร้างค่าใหม่ขึ้นมาเก็บไว้ในเครือข่ายอีเธอเรียมนี้ โดยการผสมค่ายีนนั้นจะเรียกใช้ฟังก์ชันของ Smart Contract อีกอันหนึ่งที่ชื่อ GeneScience ซึ่ง GeneScience นี้จะไม่ถูกเปิดเผยต่อสาธารณะเพื่อความสนุกกับผู้ใช้งาน และอีก Smart Contract หนึ่งคือ KittyMarket ทำหน้าที่เป็นตลาดใช้ซื้อขายแมวดิจิทัล ผู้เป็นเจ้าของสามารถประกาศขายแมวดิจิทัลบน Contract นี้ และผู้ใช้งานอื่นสามารถเลือกซื้อแมวดิจิทัลที่ได้ประกาศขายไว้

5.1.2 ส่วนติดต่อกับผู้ใช้งานที่เป็นแอปพลิเคชันมือถือระบบแอนดรอยด์

ส่วนนี้ทำหน้าที่เชื่อมต่อระหว่างผู้ใช้งานและ Smart Contract โดยมีฟังก์ชันเลี้ยงแมวดิจิทัลในหน้าหลักเพื่อให้ความบันเทิงกับผู้ใช้งาน และฟังก์ชันในการจัดการกับกระเป๋าเงิน(wallet) เช่นเดียวกับแอปพลิเคชันอื่นที่พัฒนาบนเครือข่ายอีเธอเรียมเพื่อให้ผู้ใช้งานได้รับประสบการณ์ในการใช้งานเทคโนโลยีบล็อกเชน นอกจากนั้นแล้วส่วนนี้จะเรียกใช้ฟังก์ชันของ Smart Contract ในข้อที่ 5.1.1 เช่น ผสมพันธุ์แมวดิจิทัล ซื้อและขายแมวดิจิทัล

5.2 ปัญหาและอุปสรรคที่พบระหว่างการทำงาน

5.2.1 ไลบรารี่ Nethereum ไม่เป็นที่นิยม

เนื่องจากส่วนของแอปพลิเคชันมือถือถูกพัฒนาโดย Unity ด้วยภาษา C# .Net จึงใช้ไลบรารี่ที่ช่วยเชื่อมต่อระหว่างแอปพลิเคชัน C# กับเครือข่ายอีเธอเรียมที่ชื่อ Nethereum ซึ่งไลบรารี่นี้ไม่เป็นที่นิยมเหมือนอย่าง Web3.js ซึ่งเป็นไลบรารี่ภาษา javascript จึงมีความยากในด้านของการหาวิธีการพัฒนาจากผู้ใช้ไลบรารี่คนอื่นและคู่มือที่อธิบายได้ไม่ดีมากพอ จึงใช้เวลามากในการศึกและทำความเข้าใจไลบรารี่นี้

5.2.2 ปัญหาการใช้งาน git LFS กับ Unity Project บน github

การพัฒนาแอปพลิเคชันด้วย Unity นั้นต้องจัดการกับไฟล์ที่มีประเภทที่หลากหลาย เช่น ไฟล์รูปภาพ ไฟล์เสียง ไฟล์โมเดล 3D จึงจำเป็นต้องใช้งาน git LFS (Large File Storage) ฉะนั้นไฟล์โปรเจคจึงมีขนาดใหญ่เกินกว่าที่ github อนุญาต จึงต้องใช้การจัดการ Version Control แบบใหม่อย่างเช่นเก็บไฟล์โปรเจคไว้บน Google Drive และตั้งชื่อแยกเวอร์ชั่น

5.2.3 ปัญหาความล่าช้าของเครือข่ายอีเธอเรียม

เนื่องด้วยเทคโนโลยีบล็อกเชนเป็นเครือข่ายที่มีความปลอดภัยสูง แต่มีความช้าในการรับมือกับ transaction ที่สามารถรับได้แค่ 15 transaction ต่อวินาที จึงเจอปัญหาในการออกแบบขั้นตอนการทำงานและการใช้งานของระบบ จึงไม่สามารถให้ผู้ใช้งานหยุดรอการยืนยันของ transaction ซึ่งอาจรอนานถึง 15 นาทีได้ ฉะนั้นจึงต้องออกแบบระบบการแจ้งเตือน เมื่อส่ง transaction แล้วแจ้งให้ผู้ใช้งานทราบแล้วรอการแจ้งเตือนในหน้าต่างการใช้งานอื่นแทนที่จะหยุดแอปพลิเคชันไว้ที่หน้าต่างเดิม

5.3 แนวทางการพัฒนาต่อ

ส่วนหลักของปริญญานิพนธ์นี้คือ แมวดิจิทัล ซึ่งเป็นสินทรัพย์ดิจิทัล ที่อยู่บน Smart Contract บนเครือข่ายอีเธอเรียม และด้วยคุณสมบัติของบล็อกเชน ผู้พัฒนาท่านอื่นสามารถพัฒนา Smart Contract และแอปพลิเคชันอื่นและมาใช้งานสินทรัพย์ดิจิทัลนี้ได้ เช่น สร้างเกมต่อสู้แนว MMORPG และมีระบบสัตว์เลี้ยง อาจใช้แมวดิจิทัลนี้ไปใช้งานในเกมนั้นได้ นอกจากนี้ยังสามารถสร้าง Smart Contract เพื่อเสริมการทำงานของสินทรัพย์ดิจิทัลนี้ได้ เช่น ประมูลแมวดิจิทัล , ให้เช่าเพื่อใช้ในการผสมพันธ์ , ให้เช่าเพื่อนำไปเลี้ยงดู

บรรณานุกรม

Satoshi Nakamoto , 2008 Bitcoin White paper [online]

Available : https://bitcoin.org/bitcoin.pdf

Nikhiil Rajesh , 2018 Digital Scarcity [online]

Available : https://simplesnippets.tech/what-is-digital-scarcity-is-it-useful/

Simple Contract [online]

Available : https://solidity.readthedocs.io/en/develop/introduction-to-smart-contracts.html

Harsha Goli , 2018 HD Wallet [online]

Available : https://medium.com/@harshagoli/hd-wallets-explained-from-high-level-to-nuts-and-bolts-9a41545f5b0

Willaim Young , 2009 Sustainable consumption: green consumer behaviour when purchasing products [online]

Available : https://onlinelibrary.wiley.com/doi/abs/10.1002/sd.394

Provenance , 2015 Provenance White Paper [online]

Available : https://www.provenance.org/whitepaper

Lew , 2019 ยังไม่ใช่ XRP, SCB Easy เตรียมให้บริการโอนข้ามประเทศราคาประหยัดผ่าน xCurrent ของRipple [online]

Available : https://www.blognone.com/node/109878

Ethereum Improvement Proposals (EIPs) , 2015 [online]

Available : https://eips.ethereum.org/

บรรณานุกรม (ต่อ)

CodeTract , 2017 Walkthrough of an Ethereum Improvement Proposal (EIP) [online]

Available : https://medium.com/@codetractio/walkthrough-of-an-ethereum-improvement-proposal-eip-6fda3966d171

HM land Registry [online]

Available : https://www.gov.uk/government/organisations/land-registry

Inside Magazine issue 19 , 2018 , the tokenization of asset is disrupting the financial industry, are you ready?. [online]

Available : https://www2.deloitte.com/content/dam/Deloitte/lu/Documents/financial-services/lu-tokenization-of-assets-disrupting-financial-industry.pdf

what is game engine , unity

Available : https://unity3d.com/what-is-a-game-engine

Upgradeability using Eternal Storage , openzepelin [online]

Available : https://github.com/OpenZeppelin/openzeppelinlabs/tree/master/upgradeability_using_eternal_storage

Angello Pozo , 2017 Ethereum App Architectures [online]

Available : https://medium.com/@angellopozo/ethereum-app-architectures-4add3d2c613d

Demiro Massessi , 2018 Public Vs Private Blockchain In A Nutshell [online]

Available : https://medium.com/coinmonks/public-vs-private-blockchain-in-a-nutshell-c9fe284fa39f

KC Tam , 2018 Transactions in Ethereum [online]

Available : https://medium.com/@kctheservant/transactions-in-ethereum-e85a73068f74