



Cryptonite



โดย

นาย ภาสเวียร์ เรืองสำราญ	รหัสนักศึกษา 1640700637 127D
นางสาว ชนัตถ์ กิจวานิช	รหัสนักศึกษา 1640701775 127D
นาย ชาคริส เกิดภักดี	รหัสนักศึกษา 1640703169 127D

อาจารย์ผู้สอน
ผู้ช่วยศาสตราจารย์สิรินธร จิตาศักดิ์
อาจารย์ ชัยนร์พล วุฒิวะธรรม

หลักสูตรวิทยาศาสตรบัณฑิต
ภาคการศึกษาที่ 2 ปีการศึกษา 2564
ภาควิชาวิทยาการคอมพิวเตอร์ คณะเทคโนโลยีสารสนเทศและนวัตกรรม
มหาวิทยาลัยกรุงเทพ

สารบัญ

บทที่

1. บทนำ	3
1.1 ชื่อโครงการ Cryptonite	3
1.2 วัตถุประสงค์ของโครงการ	3
1.3 ขอบเขตของโครงการ	3
2. การทำงานของแอปพลิเคชันหรือโปรแกรม	7
2.1 พังก์ชัน Register/Login	8
2.2 พังก์ชัน Profile	10
2.3 พังก์ชัน See NFT	11
2.4 พังก์ชัน Market Cap	12
2.5 พังก์ชัน Add Credit	13
2.6 พังก์ชัน Get Coin	14
2.7 พังก์ชัน Get NFT	15
2.8 พังก์ชัน Sell Coin	17
2.9 พังก์ชัน Sell NFT	18
2.10 พังก์ชัน Total Property	19
2.11 พังก์ชัน Help	20
ภาคผนวก ก Source Code (Cryptonite.py)	21
ภาคผนวก ข Module/API (BitKub)	84

บทที่ 1

บทนำ

1.1 ชื่อโครงการ

แอปพลิเคชันการจัดการการซื้อ-ขาย Cryptocurrency และ NFT (Cryptonite)

1.2 วัตถุประสงค์ของโครงการ

1. เพื่อพัฒนาแอปพลิเคชันการจัดการการซื้อ-ขาย Cryptocurrency และ NFT
2. เพื่อฝึกทักษะกระบวนการเขียนโปรแกรมด้วยภาษา Python และเป็นการใช้ความรู้เดิมและความรู้ใหม่ผนวกเข้าด้วยกันในด้านการพัฒนาโปรแกรมและฝึกทักษะกระบวนการคิดอย่างเป็นขั้นตอน มีเหตุผลและฝึกฝนการแก้ไขและรับมือกับปัญหาเฉพาะหน้าที่อาจจะต้องเผชิญขึ้นระหว่างทำโครงการ
3. เพื่อพัฒนาทักษะการทำงานเป็นทีม การให้ความร่วมมือและช่วยเหลือกันในการพัฒนาและแก้ไขปัญหาเพื่อประสิทธิภาพในการทำงานสูงสุด จำเป็นต้องใช้ความสามัคคี และการร่วมมือร่วมใจกันสร้างสรรค์ผลงานออกแบบ
4. เพื่อจำลองการเทรดคริปโตเคอร์เรนซ์ที่ใช้ค่าของเหรียญจริงจาก API ของ BitKub

1.3 ขอบเขตของโครงการ

แอปพลิเคชันการจัดการการซื้อ-ขาย Cryptocurrency และ NFT โดยจะมีการทำงานหลากหลายฟังก์ชัน โดยส่วนใหญ่จะเป็นการซื้อ-ขาย มีการบันทึกข้อมูล User เอาไว้ใน Database (DB Browser) เพื่อที่จะสามารถเรียกใช้ข้อมูลได้อย่างสะดวกและข้อมูลจะไม่สูญหายหากทำการปิดแอปพลิเคชัน เนื่องจากข้อมูลจะถูกเก็บเอาไว้ใน database และมูลค่าของเหรียญทุกฟังก์ชันจะนำข้อมูลมาจาก API ของ BitKub โดยแอปพลิเคชันนี้จะแบ่งออกเป็น 11 ฟังก์ชันการทำงานหลักได้แก่..

1) ฟังก์ชัน Register/Login

ฟังก์ชันนี้จะเป็นหน้าแรกที่ให้ User ที่เข้ามากดเลือก Register/Login

Register จะให้ User กรอก ชื่อ นามสกุล, เบอร์โทรศัพท์, Username, Password, Confirm Password และโปรแกรมจะ Generated ID จากข้อมูลที่ User กรอกเข้ามา (นำข้อมูล ชื่อ นามสกุล เบอร์โทรศัพท์ 4 digits สุดท้าย มาแปลงเป็น ID แล้วเก็บเป็น Primary Key) และจัดเก็บข้อมูลใน Database (Cryptobase.db) หน้า Login จะให้ User กรอก Username โดยจะดึงข้อมูลมาจาก Database ฟังก์ชัน Login สามารถเรียกใช้ได้จาก Menu bar แต่หากเมื่อทำการ Login ไปแล้ว เมื่อ กดใช้เมนูบาร์ระบบจะถามว่าต้องการ Log Out หรือไม่หากกด Yes ระบบจะทำการ Log Out และกลับไปสู่หน้า Login

2) ฟังก์ชัน Profile

ฟังก์ชันนี้จะเป็นหน้าที่สองหลักจาก User ทำการ Login เข้ามา

Profile จะแสดง ชื่อ นามสกุล ของ User และจำนวนเหรียญที่ User มีอยู่สามารถเลือกได้จาก OptionMenu และระบบจะแสดงเงินบาทที่ User มี และมีปุ่ม See NFT ที่ สามารถ กดเข้าไปดู NFT ที่ User มี และมีปุ่ม Add Credit, Marget Cap, Total Property โดยข้อมูลทั้งหมดนี้จะถูกดึงมากจาก Database (Cryptobase.db) และทุกฟังก์ชันจะมีข้อมูลของ User ปรากฏอยู่บน root.title ฟังก์ชันนี้สามารถเรียกใช้ได้จาก Menu bar แต่ถ้าหากยังไม่ได้ Login ระบบจะไม่อนุญาตให้เข้าถึงหน้านี้

3) ฟังก์ชัน See NFT

ฟังก์ชันนี้จะแสดง NFT ที่ User มีอยู่ โดยสามารถเข้าฟังก์ชันนี้ได้จากหน้า Profile โดยฟังก์ชันนี้ จะแสดง NFT ของ User โดยการดึงข้อมูล NFT ที่ User มีมาจาก Database (cryptobase.db) Table user_data ถ้าหากระบบเจอบข้อมูล NFT ของ User ระบบจะนำมาแสดง ฟังก์ชันนี้สามารถ เรียกใช้ได้จาก Menu bar แต่ถ้าหากยังไม่ได้ Login ระบบจะไม่อนุญาตให้เข้าถึงหน้านี้

4) พังก์ชัน Market Cap

พังก์ชันนี้จะแสดงราคาของแต่ละเหรียญ Crypto ว่ามีราคายุ่งที่เท่าไหร่ (BTC, ETH, DOGE, KUB, BNB) เพื่อเป็นการประกอบการตัดสินใจในการซื้อของ User โดยมูลค่าของเหรียญนั้นมาจากการดึงข้อมูลมาจาก API ของ BitKub (API_HOST = "https://api.bitkub.com") พังก์ชันนี้สามารถเลือกใช้ได้จาก Menu bar แต่ถ้าหากยังไม่ได้ Login ระบบจะไม่อนุญาตให้เข้าถึงหน้านี้

5) พังก์ชัน Add Credit

พังก์ชันนี้เป็นพังก์ชันการเติมเงินบาท เพื่อนำเงินบาทไปซื้อเหรียญ Crypto User สามารถเลือกจำนวนเงินบาทที่จะเติมเข้าแอปพลิเคชัน โดยจะมีให้เลือกเติม 500 บาท, 1000 บาท, 3000 บาท และ Enter amount โดยระบบสามารถให้ User กรอกจำนวนเงินที่ต้องการเติมได้ด้วยตนเอง เมื่อเติมเงินเสร็จแล้ว ระบบจะทำการส่งข้อมูลไปยัง Database (Cryptobase.db) และจะแสดงจำนวนเงินล่าสุดทุกครั้ง พังก์ชันนี้สามารถเลือกใช้ได้จาก Menu bar แต่ถ้าหากยังไม่ได้ Login ระบบจะไม่อนุญาตให้เข้าถึงหน้านี้

6) พังก์ชัน Get Coin

พังก์ชันนี้เป็นพังก์ชันซื้อเหรียญ โดยจะมีเหรียญให้เลือกหลากหลายชนิด โดย User ต้องใช้เงินบาทในการซื้อเหรียญ และมูลค่าของเหรียญถูกดึงมาจาก API ของ Bitkub เมื่อ User ซื้อเหรียญเรียบร้อยแล้ว ระบบจะทำการบันทึก แล้วจะแสดงเหรียญที่ User ถืออยู่ ในหน้า Profile พังก์ชันนี้สามารถเลือกใช้ได้จาก Menu bar แต่ถ้าหากยังไม่ได้ Login ระบบจะไม่อนุญาตให้เข้าถึงหน้านี้

7) พังก์ชัน Get NFT

พังก์ชันนี้จะเป็นพังก์ชันที่มีไว้สำหรับการซื้อ NFT โดย User จะต้องทำการแลกซื้อ Near ก่อนเพื่อจะนำ Near ไปซื้อ NFT โดยระบบจะให้เลือกจำนวน Near ที่ต้องการแล้วจะคำนวณเป็นเงินบาทให้เพื่อประกอบการตัดสินใจกับค่าใช้จ่ายที่จะต้องจ่าย ระบบจะทำการตรวจสอบว่าเงินของ User พอดีจะซื้อ Near หรือไม่ถ้าหากไม่พ่อระบบจะแนะนำให้ไปเติมเงินที่พังก์ชัน Add Credit ก่อน โดยสามารถดูราคา NFT ได้โดยกดที่รูปที่ต้องการซื้อ ระบบจะแสดงรูป, ผู้สร้าง, ราคา Near ราคาเงินบาท ที่เฟรมผ่องขาวเพื่อประกอบการตัดสินใจ หากรูปไหนถูกซื้อไปแล้วระบบจะแจ้งว่ารูปนั้น Sold

Out จะไม่สามารถซื้อด้วย เมื่อกดซื้อ NFT ระบบจะตรวจสอบ Near ว่าเพียงพอหรือไม่ และเมื่อซื้อเสร็จระบบจะหัก Near จาก Database (Cryptobase.db) ตามจำนวนที่กำหนดไว้ และรูปภาพจะแสดงว่า Sold Out และไม่สามารถกดซื้อด้วยอีก พังก์ชันนี้สามารถเลือกใช้ได้จาก Menu bar แต่ถ้าหากยังไม่ได้ Login ระบบจะไม่อนุญาตให้เข้าถึงหน้านี้

8) พังก์ชัน Sell Coin

พังก์ชันนี้คือพังก์ชันซื้อขายเหรียญ โดย User สามารถเลือกเหรียญแล้วระบบจะแสดงจำนวนเหรียญที่เรามี User สามารถเลือกจำนวนเหรียญที่จะขายด้วย และจะมีช่องแสดงราคาที่เราจะได้จากการขายเหรียญ เมื่อกดขายเหรียญระบบจะทำการตรวจสอบทุกครั้งว่า User มีเหรียญพอ กับจำนวนที่ต้องการขายจริงหรือไม่ ถ้าหากมีจำนวนเหรียญไม่พอระบบจะแนะนำให้ User ไปซื้อเหรียญที่พังก์ชัน Get Coin เมื่อทำการขายเหรียญ เสิร์ฟเวอร์อยู่ระบบจะทำการ Update ข้อมูลไปยัง Database (Cryptobase.db) ทุกครั้ง และพังก์ชันนี้สามารถเลือกใช้ได้จาก Menu bar แต่ถ้าหากยังไม่ได้ Login ระบบจะไม่อนุญาตให้เข้าถึงหน้านี้

9) พังก์ชัน Sell NFT

พังก์ชันนี้จะเป็นพังก์ชันการขาย NFT โดยเมื่อ User ทำการกดเข้าไปถ้าระบบตรวจสอบแล้วพบว่า User มีรูปภาพที่ต้องการขาย ระบบจะแสดงรูปที่ User มีเพื่อที่สามารถกดขายได้ User สามารถกดที่รูปที่ต้องการขายแล้วระบบจะแสดงข้อมูลที่เฟรมฝั่งขวาเมื่อ โดยจะแสดงรูป, ชื่อผู้สร้าง, ค่า Near, และราคาเงินบาทที่ User จะได้รับเป็นผลตอบแทนจากการขาย เมื่อกด Sell NFT ระบบจะแสดงค่า Near และต่อมาก็แสดงจำนวนเงินที่จะได้ เมื่อ User กด Yes ระบบจะทำการขายรูปนั้น และ User จะได้รับเงินบาท ระบบจะไม่แสดงรูปนั้นในพังก์ชัน See NFT เนื่องจาก User ขายไปแล้ว เมื่อทำการขาย NFT เสิร์ฟเวอร์อยู่ระบบจะทำการ Update ข้อมูลไปยัง Database (Cryptobase.db) ทุกครั้ง พังก์ชันนี้สามารถเลือกใช้ได้จาก Menu bar แต่ถ้าหากยังไม่ได้ Login ระบบจะไม่อนุญาตให้เข้าถึงหน้านี้

10) พังก์ชัน Total Property

พังก์ชันนี้จะเป็นพังก์ชันแสดงสินทรัพย์รวมทั้งหมดในแอพพลิเคชันนี้ โดยจะนำข้อมูลสินทรัพย์ที่ User มีทั้งหมดมารวมกัน โดยจะนำมาจาก THB (เงินบาทที่ User มี), จำนวนเหรียญที่ User มี (นำทุกเหรียญที่ User มีมาแปลงเป็นราคาเงินบาท), NFT (นำ NFT ทั้งหมดมาแปลงเป็นเงินบาท) และระบบจะแสดงจำนวนเงินออมมา โดยข้อมูลทั้งหมดถูกนำมาจาก Database (Cryptobase.db)
พังก์ชันนี้สามารถเลือกใช้ได้จาก Menu bar แต่ถ้าหากยังไม่ได้ Login ระบบจะไม่อนุญาตให้เข้าถึงหน้านี้

11) พังก์ชัน Help

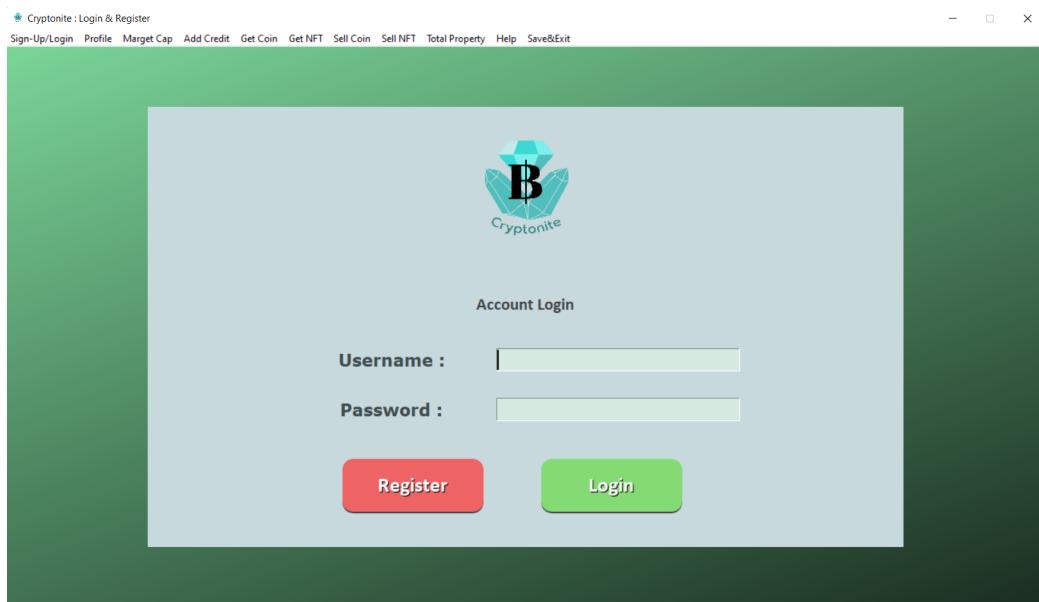
พังก์ชันนี้จะเป็นพังก์ชันแสดงวิธีการใช้แอพพลิเคชันนี้ โดยจะแสดงเป็น Cheat Sheet พังก์ชันนี้สามารถเข้าถึงได้เลย ไม่จำเป็นต้อง Login หรือ Register ก่อน จุดประสงค์เพื่อจะให้ User สามารถรู้วิธีการใช้งานทั้งหมด โดยคร่าว ๆ ว่าพังก์ชันต่าง ๆ มีหน้าที่ทำอะไรบ้าง

บทที่ 2

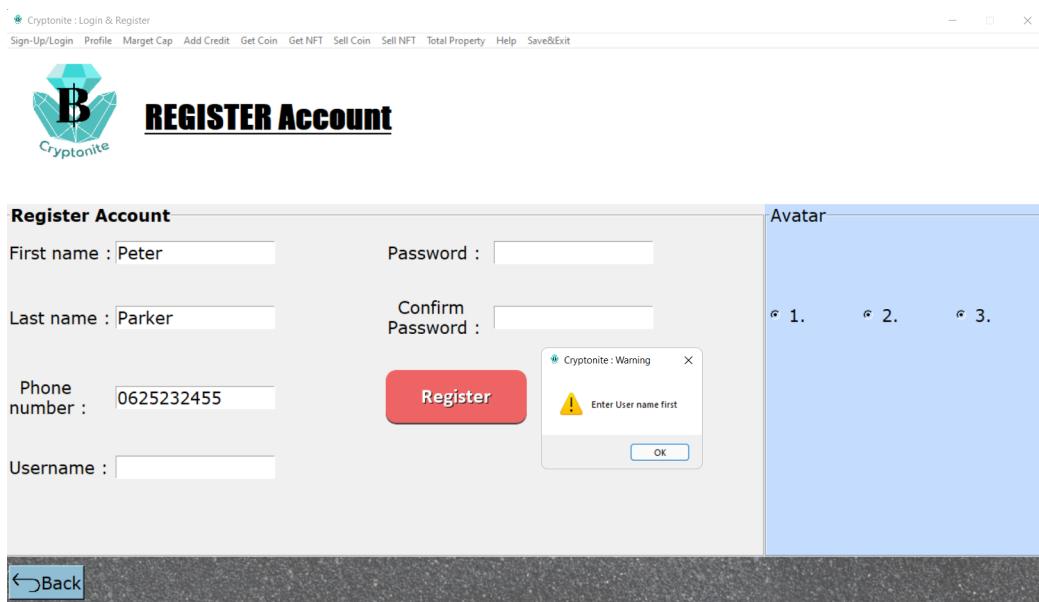
การทำงานของแอปพลิเคชัน

2. พัฒนาการทำงานของแอปพลิเคชัน Cryptonite

2.1) หน้าฟังก์ชัน Register/Login



ภาพที่ 2.1.1 Login



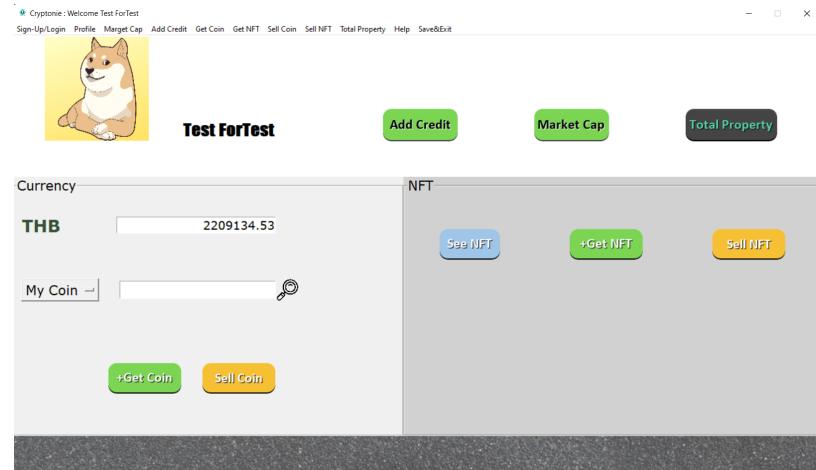
ภาพที่ 2.1.2 Register

คำอธิบาย

2.1.1) เมื่อเปิดโปรแกรมจะเจอนหน้า Login เป็นอันดับแรก ให้ User ทำการ Login ในกรณีที่ User ยังไม่มี Username และ Password ให้ทำการกด Register เพื่อทำการสมัคร ในกรณีที่ User ไม่ได้ทำการ Login จะทำให้ User ไม่สามารถเรียกใช้ฟังก์ชันใน Menubar ได้ โดยมีระบบหลังบ้านที่จะคุยกับข้อมูลที่ User กรอกไว้ใน Database (Cryptobase.db) Table user_data หรือไม่ถ้าหากไม่พบรอบจะแจ้ง Messagebox ว่าให้กรอกรหัสผ่านใหม่

2.1.2) เมื่อกด Register เข้ามาโปรแกรมจะให้ User ทำการกรอกข้อมูลส่วนตัวเพื่อทำการสมัคร ในกรณีที่ User ใส่ข้อมูลไม่ครบทางโปรแกรมจะขึ้นแจ้งเตือนและให้ User กรอกข้อมูลของช่องนั้น ๆ โดยข้อมูลที่ระบบต้องการจะให้กรอก First name, Last name, Phone number, Username, Password, Confirm Password และเลือก Avatar ต่อมาสามารถกด Back เพื่อกลับไปยังหน้า Login โดยระบบหลังบ้าน จะคุยกับข้อมูลได้ถูกกรอกทุกช่องหรือไม่โดยจะมีการเช็คเงื่อนไขภายในโค้ด และเมื่อกรอกครบ User จะต้องเลือก Avatar โดยจะมี 3 รูปให้เลือก 1.Dogey 2.Mr.increed 3.Alien โดยทั้งหมดหลังบ้านจะมีการเช็คอยู่เสมอว่าข้อมูลนั้นถูกใช้ไปแล้วหรือยัง เมื่อเช็คเรียบร้อยว่าข้อมูลไม่ซ้ำ ระบบจะทำการ Generate ID จากข้อมูลที่ได้จากการกรอกไปเป็น Primary Key ในช่อง id ใน Database (Cryptobase.db) Table user_data และจะ set ค่า เหรียญต่างๆ เป็น 0 และมีการให้เงิน User สำหรับเริ่มต้นการใช้งาน 500 THB

2.2) พังก์ชัน Profile

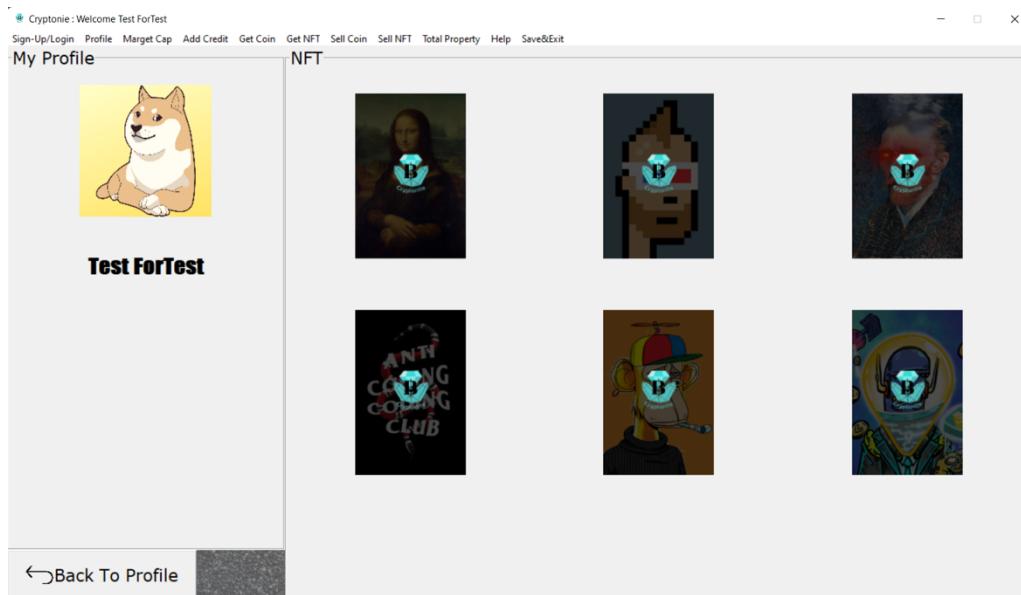


ภาพที่ 2.2.1 Profile

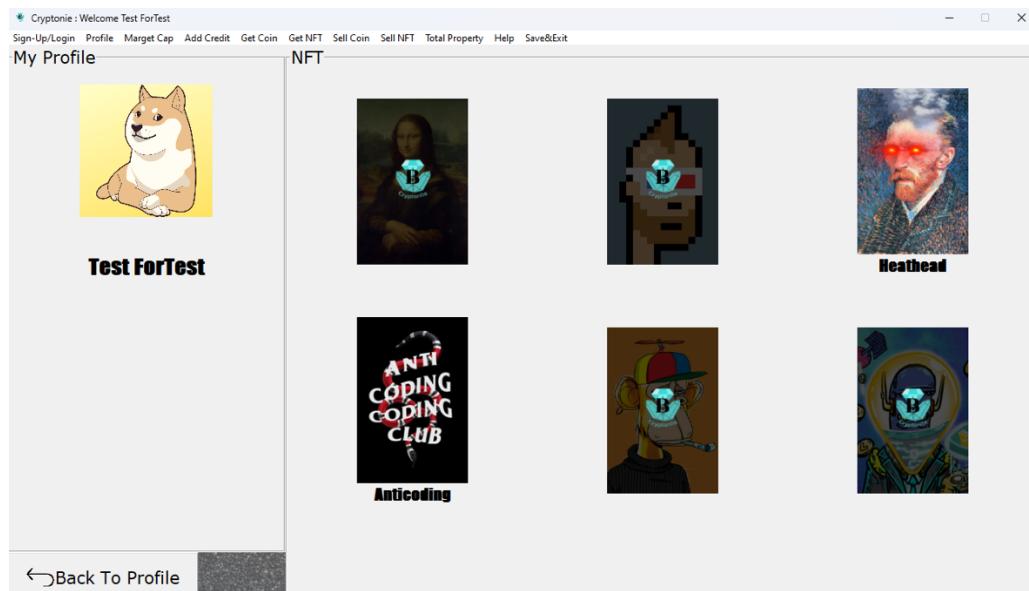
คำอธิบาย

2.2.1) เมื่อ User เลือกพังก์ชัน Profile ระบบจะแสดงข้อมูลของ User ที่ root.title และในเฟรม และระบบจะแสดงข้อมูลต่าง ๆ ของ User เช่น จำนวนเงินบาท จำนวนเหรียญ และจะมีปุ่มต่าง ๆ เพื่อกดไปยังพังก์ชันอื่น และจะมีปุ่ม See NFT ที่จะพาไปยังพังก์ชัน See NFT ที่จะสามารถดู NFT ที่ User มีได้ ในส่วนของหลังบ้านระบบจะมีการดึงข้อมูลมาจาก Database (Cryptonite.db) Table user_data เพื่อนำข้อมูลมาใช้ในการแสดงข้อมูลต่าง ๆ เช่นช่อง THB จะดึงมาจาก Table user_data ช่อง money โดยดึงมาเก็บใน result ข้อมูลนี้จะสามารถเรียกได้โดยมีชื่อว่า result[7] ส่วนเหรียญจะเป็นข้อมูลที่กดหาได้ใน OptionMenu เมื่อ User กดหา ระบบหลังบ้านจะทำการตรวจสอบใน Table user_data ว่า User นี้มีเหรียญเท่าไหร่และจะนำมาแสดงผลใน Entrybox

2.3) พังก์ชัน See NFT



ภาพที่ 2.3.1 See NFT



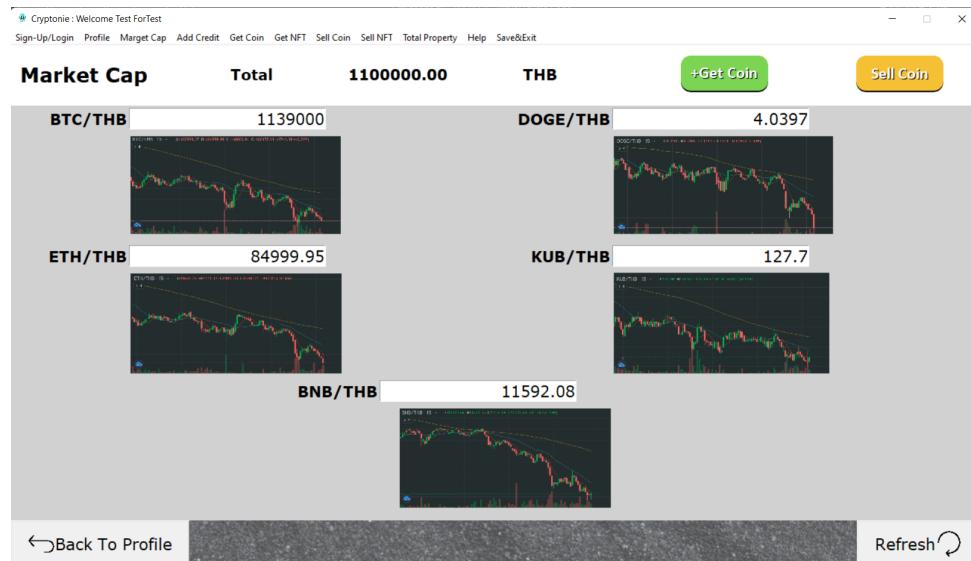
ภาพที่ 2.3.2 See NFT

คำอธิบาย

2.3.1) เมื่อ User กดเข้ามาฟังก์ชัน See NFT จากปุ่มในฟังก์ชัน Profile เมื่อเข้ามาแล้วระบบจะแสดงชื่อของ User บน root.title ต่อมาระบบจะแสดงรูปภาพ Avatar ใน LabelFrame My Profile ฝั่งซ้าย และชื่อผู้ใช้งาน ในส่วนฝั่งขวาจะแสดง NFT ในฝั่งขวาของ LabelFrame NFT โดยจะแสดง NFT ที่ User มี ถ้าหากว่า User ไม่ได้ซื้อเอาไว้ ระบบจะขึ้นรูปภาพเบลอและโลโก้ของแอพพลิเคชัน ดังภาพที่ 2.3.2 ระบบหลังบ้านจะมีการตรวจสอบข้อมูลว่า User มี NFT ลันไหนบ้าง

และจะนำมาแสดง โดยเช็คจาก Table user_data โดย NFT จะจัดเก็บเป็นช่องของตัวเองอันละ 1 ช่อง ระบบจะเช็คผ่านเงื่อนไข if และวิจัยนำรูปที่ User มีจริง ๆ นำมาแสดงผลในฟังก์ชันนี้

2.4) ฟังก์ชัน Market Cap

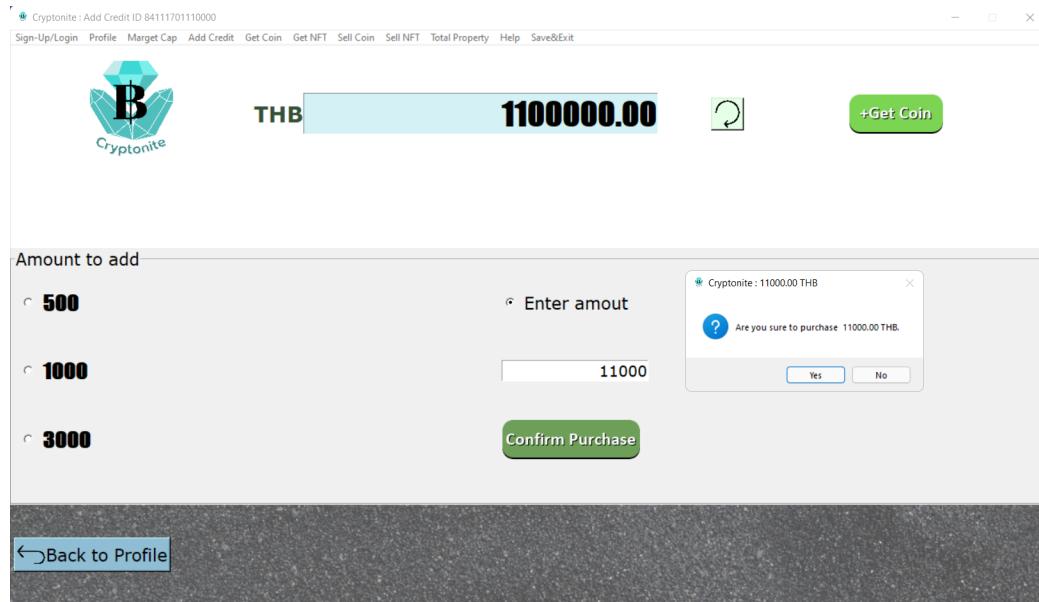


ภาพที่ 2.4 Market Cap

คำอธิบาย

2.4) ในหน้าฟังก์ชัน Market Cap คือฟังก์ชันให้ User ดูค่าเงินของเหรียญต่าง ๆ เพื่อประกอบการตัดสินใจของ User โดยค่าเงินที่ปรากฏเป็นค่าเงินที่ดึง API มาจากเว็บ <https://api.bitkub.com> ค่าเงินสามารถแปรผันตามความจริง โดยนำข้อมูลมา 5 เหรียญ 1.BTC (Bitcoin) 2.ETH (Ethereum) 3.DOGECOIN (Dogecoin) 4. KUB (BitKub) 5.BNB (Binance) นำมาเป็นค่าเงินบาทไทย และสามารถกดปุ่ม Refresh เพื่อ Update ราคาเหรียญ มีปุ่ม Back To Profile เพื่อกลับไปยังหน้าprofile หรือไปยังฟังก์ชัน Profile และปุ่มอื่นๆ เพื่อไปยังฟังก์ชันต่าง ๆ ระบบหลังบ้านคือการดึงข้อมูล API มาจากเว็บ BitKub <https://api.bitkub.com/api/market/ticker> Module requests

2.5) พื้นที่ Add Credit

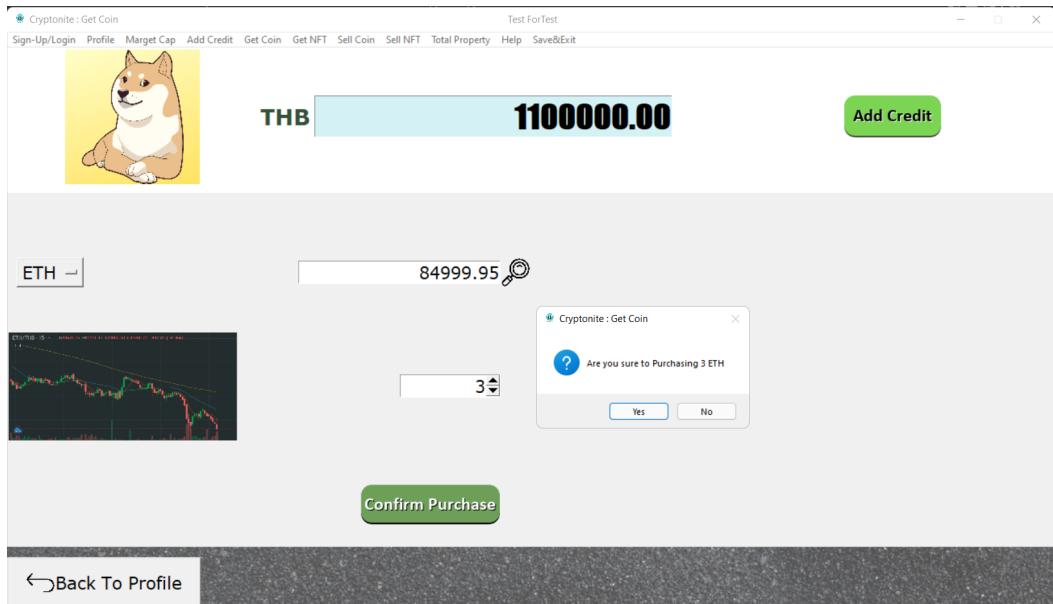


ภาพที่ 2.5 Add Credit

คำอธิบาย

2.5) พื้นที่ Add Credit คือพื้นที่ในการทำงานที่ให้ User ทำการเติมเงินบาทเข้าไปในบัญชีของตนเอง โดยสามารถเลือกจำนวนเงินทางด้านซ้ายหรือสามารถกรอกจำนวนเงินตามที่ user ต้องการได้ กดปุ่ม Comfirm Purchase เพื่อเป็นการยืนยันการเติมเงิน เมื่อยืนยันแล้วเงินจะเข้าบัญชี อัตโนมัติ สามารถกดปุ่ม Back to Profile เพื่อกลับไปหน้า Profile ได้ และยังมีปุ่ม Get Coin เพื่อกดแล้วไปยังพื้นที่ Get Coin ได้ และในส่วนของหลังบ้าน เมื่อ User เลือกจำนวนเงินเสร็จ ระบบจะเพิ่มเงินไปยัง Database (Cryptobase.db) Table user_data ช่องที่ 7

2.6) พังก์ชัน Get Coin

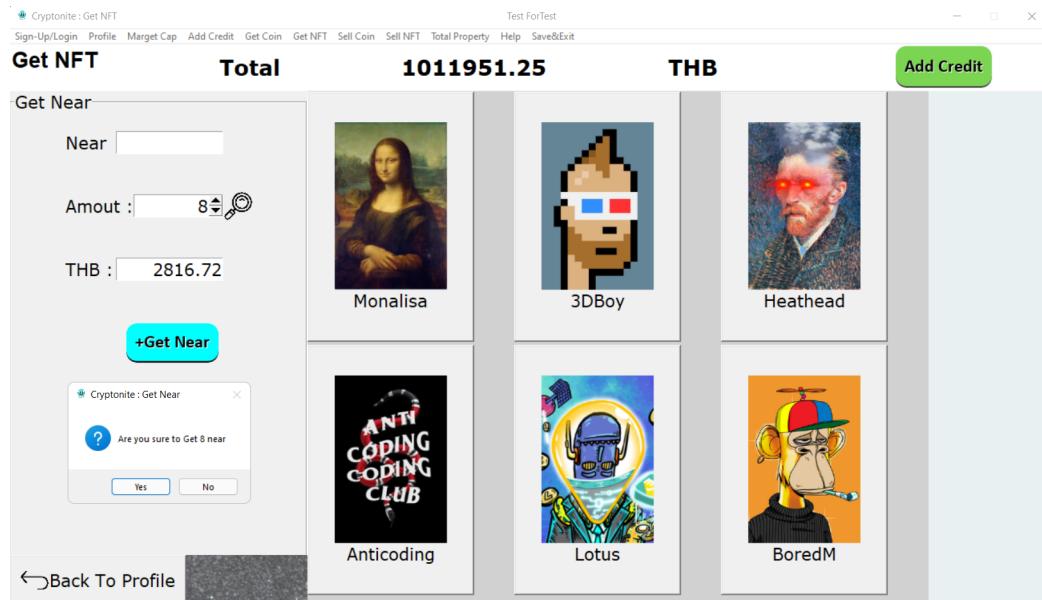


ภาพที่ 2.6 Get Coin

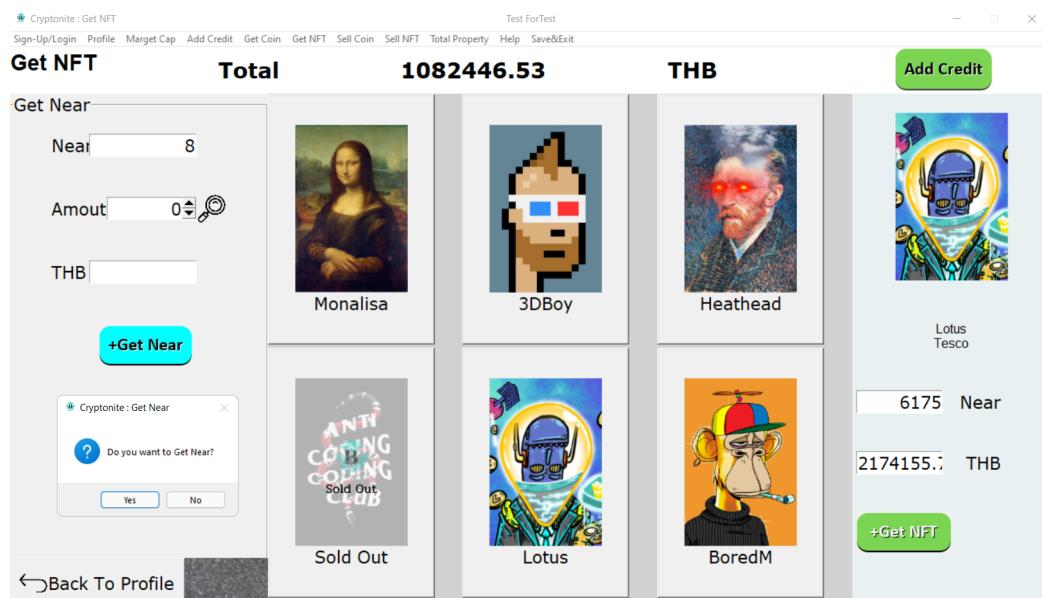
คำอธิบาย

2.6) ในหน้าGet Coin userสามารถค้นหาหรือค่าเงินและเลือกซื้อหรือขายในจำนวนที่ต้องการได้เมื่อ user กดปุ่ม Confirm Purchase ทางโปรแกรมจะแสดง Messagebox เพื่อให้ user ยืนยันการซื้อขายครั้งนี้ครั้ง โดยถ้า User ยืนยันระบบจะแสดงจำนวนเงินที่ user ต้องจ่ายและให้ user ยืนยันอีกครั้งหลังจากยืนยันแล้วระบบจะเพิ่มหรือลด User และทำการหักเงินออกจากบัญชีของ User แต่ในกรณีที่ User ไม่ยืนยันระบบจะแจ้งเตือนว่า User ได้ทำการยกเลิกการซื้อ ในส่วนของหลังบ้าน ระบบจะมีหรือไม่มีให้เลือกใน OptionMenu User สามารถเลือกหรือไม่เลือกที่ต้องการซื้อได้ เมื่อเลือกแล้วกดปุ่ม Search ระบบจะดึงค่าหรือราคามาจาก API ของหรือจากเว็บไซต์ ฯ และจะแสดงราคา User สามารถปรับเพิ่มลดหรือเปลี่ยนได้ เมื่อเลือกจำนวนเสร็จแล้วระบบจะคำนวณราคากลางๆ ตามจำนวนที่ซื้อ

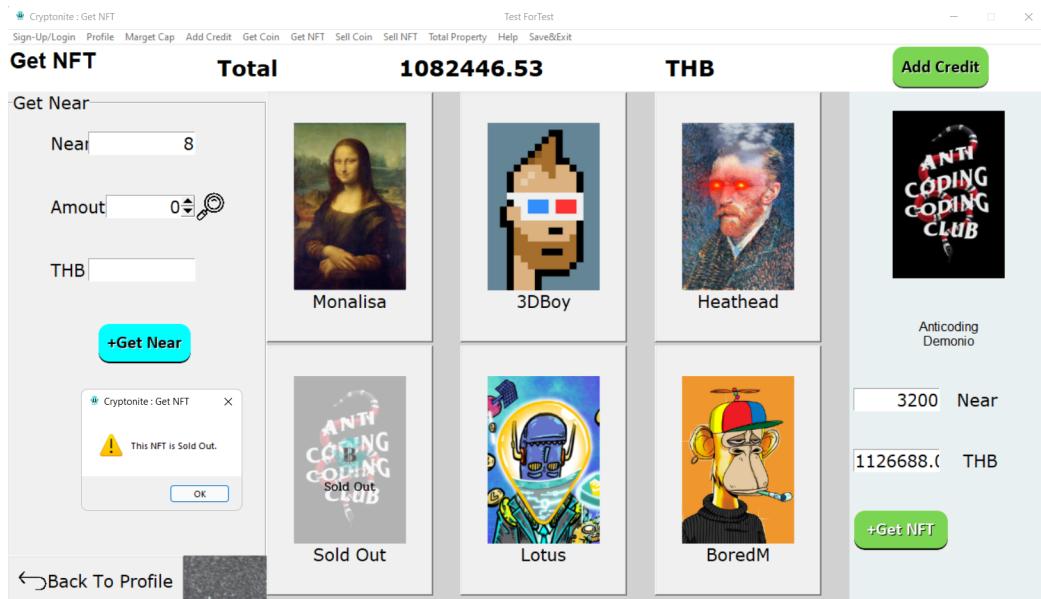
2.7) พังก์ชัน Get NFT



ภาพที่ 2.7.1 Get NFT



ภาพที่ 2.7.2 Get NFT



ภาพที่ 2.7.3 Get NFT

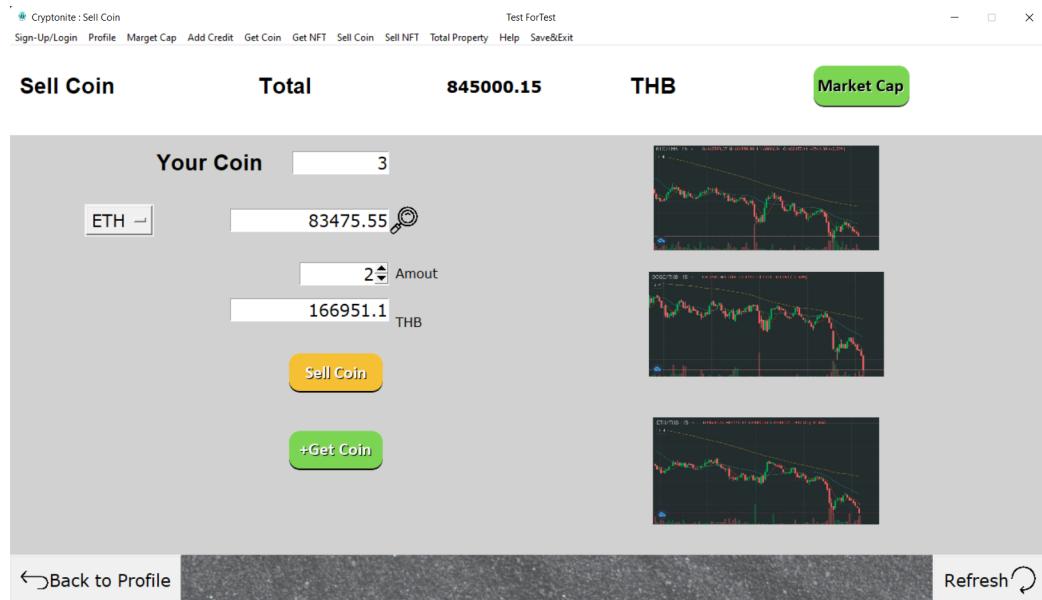
คำอธิบาย

2.7.1) ในหน้าฟังก์ชันGet NFT User จะสามารถซื้อภาพ NFT ได้ด้วยเหรียญ Near โดย User สามารถซื้อเหรียญ Near ได้โดยการใส่จำนวนเหรียญที่ต้องการซื้อ ทางโปรแกรมจะบอกราคาที่ User ต้องจ่าย เมื่อกดปุ่ม Get Near โปรแกรมจะแสดง Messagebox ให้ User ยืนยันการซื้อเหรียญ ในกรณีที่ภาพอ่อน ๆ ถูกซื้อไปแล้วจะแสดงว่ารูปภาพนั้นถูกซื้อไปแล้ว ระบบหลังบ้านคือ จะมีค่าเงิน Near อุยกะที่ 352.09 ระบบจะคำนวนตามจำนวนที่ User ต้องการซื้อเหรียญ และเมื่อซื้อเสร็จแล้ว ระบบจะเพิ่มเหรียญ Near ไปยัง Database (Cryptobase.db) Table user_data ช่องที่ 20 เพื่อเก็บเหรียญ Near และนำไปซื้อ NFT ในลำดับถัดไป

2.7.2) ในกรณีที่ User ต้องการซื้อรูปภาพ NFT แต่จำนวน Near ของ User ไม่พอ ทางโปรแกรมจะแสดง Messagebox เพื่อสอบถาม User ในการเติม Near ในกรณีที่ User ต้องการซื้อ Near ระบบจะไปสู่หน้าซื้อ Near โดยอัตโนมัติ ในส่วนของระบบหลังบ้าน จะมีการตรวจสอบว่า เหรียญ Near มีเพียงพอหรือไม่เพื่อที่จะซื้อ NFT ถ้าหากว่าไม่เพียงพอระบบจะมีการตรวจสอบใน Database ว่าไม่พอจริงหรือไม่

2.7.3) ในกรณีที่รูปภาพ NFT ถูกซื้อไปแล้ว ทางระบบจะแสดงว่ารูปภาพนี้ถูกซื้อไปแล้วและไม่สามารถซื้อรูปภาพได้ หลังบ้านคือ ระบบจะตรวจสอบว่ารูปนี้ได้ถูกซื้อไปหรือยังจาก Database

2.8) พื้นที่ชั้น Sell Coin

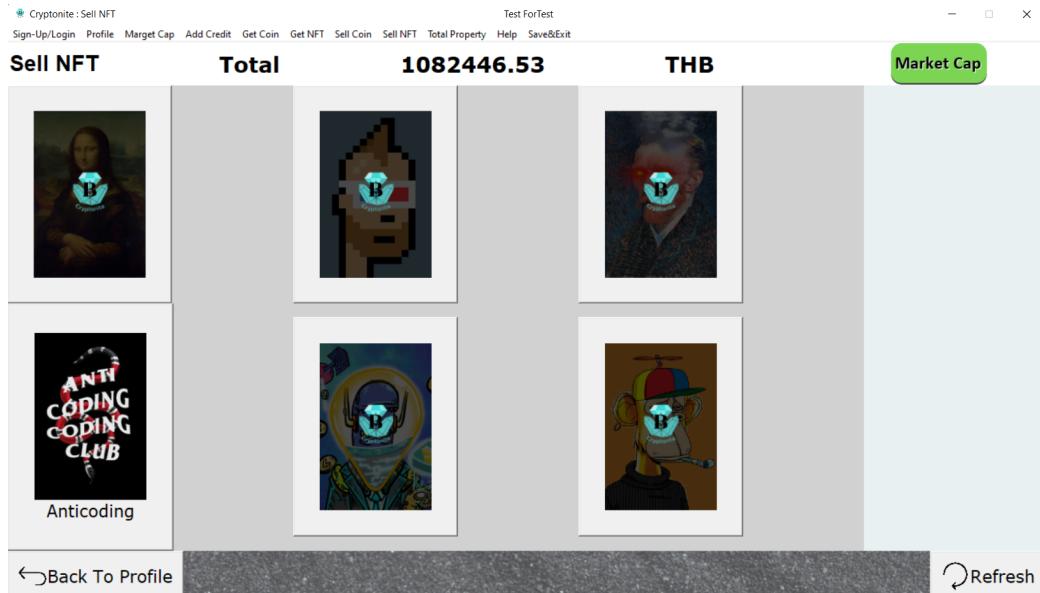


ภาพที่ 2.8 Sell Coin

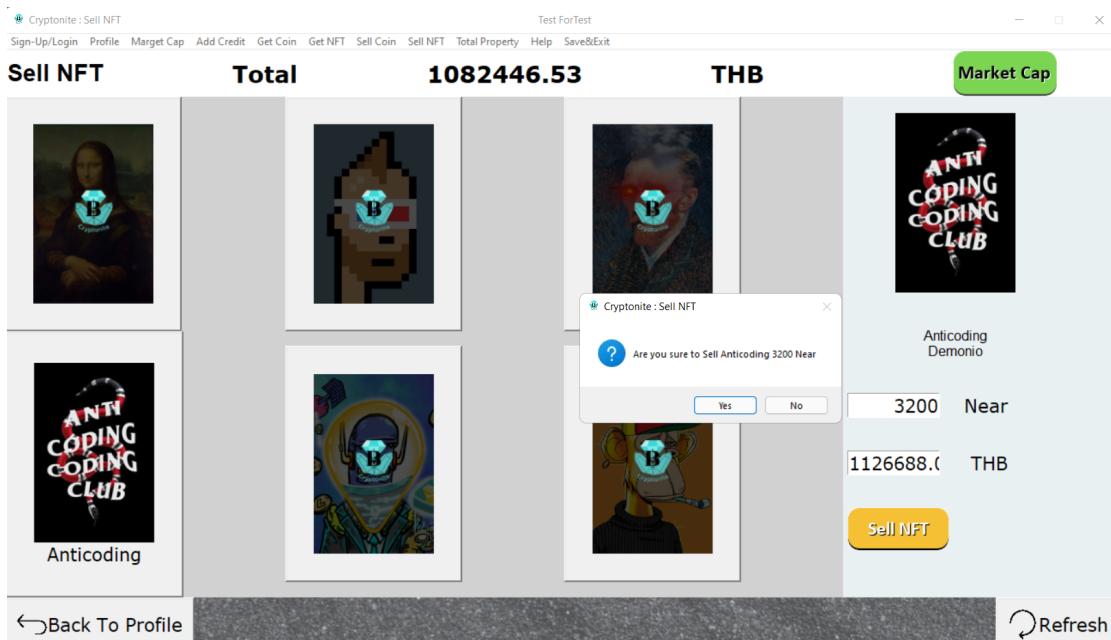
คำอธิบาย

ในหน้า Sell Coin user สามารถเลือกเหรียญที่ต้องการจะขายและเมื่อกดปุ่มคันหาจะแสดงค่าเงินของเหรียญนั้นและแสดงจำนวนเงินที่ User จะได้หลังจากขายเหรียญ เมื่อกดขายเหรียญโปรแกรมจะแสดง Messagebox ให้ User ยืนยันการขายเหรียญ หลังจากขายเหรียญสำเร็จเงินจะถูกเพิ่มเข้าไปในบัญชีของ User ระบบหลังบ้านคือจะเช็คเหรียญของ User ว่ามีเหรียญจริงหรือไม่ เมื่อเช็คเรียบร้อยระบบจะคำนวณอุปกรณามาเป็นจำนวนเงินที่ User จะได้รับ โดยดึงจำนวนเหรียญที่ User มีมา จาก Database และนำมาคำนวณกับค่าเหรียญของตอนนั้น และเมื่อข้อสำเร็จระบบจะนำเงินที่ User ได้รับมาจากการขายเหรียญ นำไปเพิ่มใน Database และหักจำนวนเหรียญที่ขายไปออก

2.9) พังก์ชัน Sell NFT



ภาพที่ 2.9.1 Sell NFT



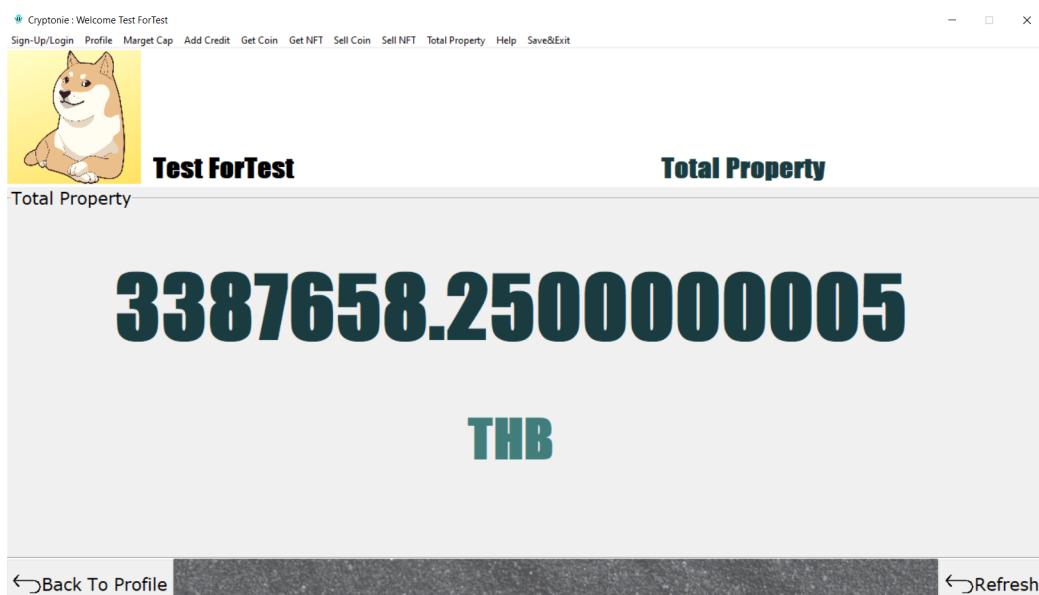
ภาพที่ 2.9.2 Sell NFT

คำอธิบาย

2.9.1) ในพังก์ชัน Sell NFT ระบบจะแสดงภาพที่ User มีให้ซัดเจน ในกรณีที่ User ไม่มีภาพ NFT นั้นๆ รูป NFT จะทำการเบลอและทำให้ไม่สามารถขายภาพ NFT นั้นได้ และระบบจะทำการแจ้งเตือนสาม User ว่าต้องการซื้อภาพ NFT หรือไม่ต้องการ ถ้า User ต้องการระบบจะทำการเด้งไปหน้า Get NFT อัตโนมัติ ระบบหลังบ้านจะตรวจสอบ NFT ของ User จาก Data Base

2.8.2) ในกรณีที่ User ต้องการขายภาพ NFT ที่ User มีรูปแบบจะแจ้งเตือนจำนวนเหรียญ Near และแปลงเป็นเงินบาทที่ User จะได้รับหลังจากที่ User ยืนยันขายภาพ NFT ในส่วนของหลังบ้าน เนื่องจากการซื้อขายเป็นที่เรียบร้อย ระบบจะทำการ Update ข้อมูลไปยัง Database (Cryptobase.db) Table user_data จะทำการเพิ่มเงินที่ได้จากการขาย NFT และนำรูปภาพออกจาก Database ของ User และระบบจะไม่แสดงรูปภาพของ User อีกเนื่องจากได้มีการขายออกไปแล้ว

2.10) พังก์ชัน Total Property

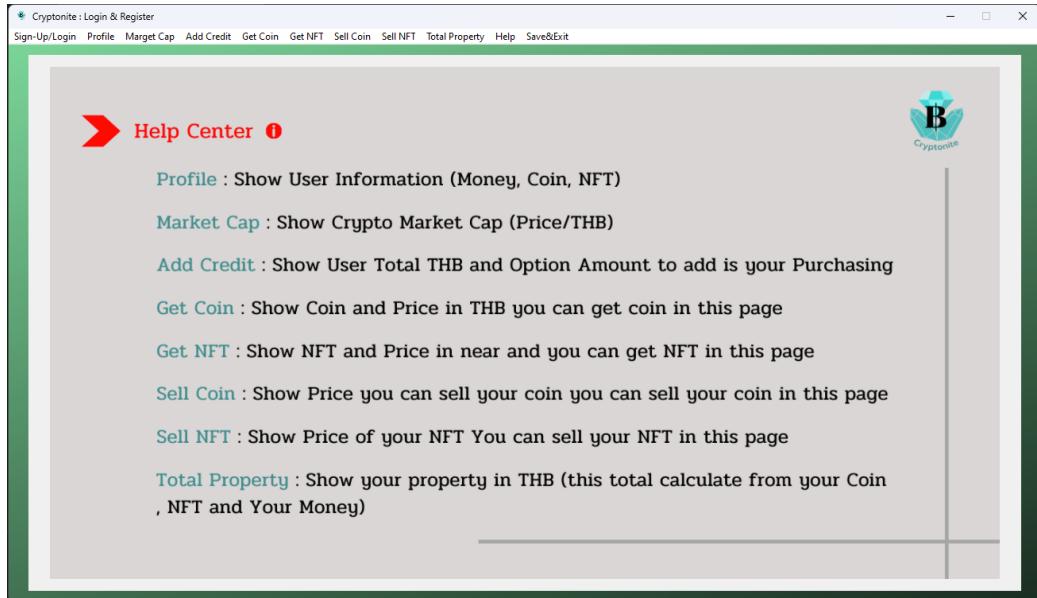


ภาพที่ 2.10 Total Property

คำอธิบาย

2.10) ในหน้าพังก์ชันนี้จะแสดงจำนวนเงินทั้งหมดที่ User มีในระบบเป็นค่าเงินบาท ทุกสินทรัพย์ ที่ User มี หรือครอบครองไว้จะถูกนำมารวม และแสดงเป็นจำนวนเงินบาทในหน้านี้ ระบบหลังบ้าน จะมีการดึงข้อมูลมาจาก Database (cryptobase.db) Table user_data เช่น จำนวนเงินบาทที่ User มี, จำนวนเหรียญที่ User มีไว้ในครอบครองนำมาแปลงเป็นเงินบาท, NFT นำมาแปลงเป็น ราคาขาย และนำทั้งหมดมารวม และแสดงในหน้านี้

2.11) พังก์ชัน Help



ภาพที่ 2.11 Help

คำอธิบาย

2.11) ในพังก์ชัน Help เป็นพังก์ชันช่วยอธิบายการใช้งานของฟังก์ชันต่างๆไว้ เพื่ออำนวยความสะดวกให้แก่ user บางท่านที่ไม่เข้าใจในการทำงานของแต่ละฟังก์ชัน โดยฟังก์ชันนี้ไม่จำเป็นต้อง Login เข้ามา ก็สามารถเปิดได้ ระบบหลังบ้านจะยังไม่นับว่าเข้าไปอยู่ร่วมกับระบบอื่น ๆ ที่ต้อง Login ก่อน เพื่อเป็นการอธิบายก่อนเริ่มใช้งาน ในรูปแบบของ Cheat Sheet

ການພັດງານ ອີ Source Code (Cryptonite.py)

Source Code

```
#-----
#Project : Cryptonite
#Create by
#   Haris Kirdpakdee    1640703169
#   Chanut Kidwat       1640701775
#   Pasawe Ruangsamran 1640700637
#Section : 127D
#-----

import Module
from tkinter import*
import sqlite3
from tkinter import ttk
from tkinter import messagebox
import requests
API_HOST = "https://api.bitkub.com"

#Create Main Window
def mainwindow() :
    root = Tk()
    x = root.winfo_screenwidth()/2 - w/2
    y = root.winfo_screenheight()/2 - h/2
    root.geometry("%dx%d+%d+%d"%(w,h,x,y))
    root.config(bg='#EBFCE0')
    root.title("Cryptonite : Login & Register")
    root.option_add("*font", "Verdana 16 bold")
    menubar = Menu(root)
    menubar.add_command(label="Sign-Up/Login", command=backtologin)
    menubar.add_command(label="Profile", command=profileClick)
    menubar.add_command(label="Marget Cap", command=margetcap_fn)
    menubar.add_command(label="Add Credit", command=addcredit_fn)
    menubar.add_command(label="Get Coin", command=getcoin_fn)
    menubar.add_command(label="Get NFT", command=getnft_fn)
    menubar.add_command(label="Sell Coin", command=sellcoin_fn)
    menubar.add_command(label="Sell NFT", command=sellnft_fn)
    menubar.add_command(label="Total Property",
    command=totalproperty_fn)
    menubar.add_command(label="Help", command=help_fn)
    menubar.add_command(label="Save&Exit", command=exit_fn)
    root.config(menu=menubar)
    root.resizable(False, False)
    root.rowconfigure((0,1,2,3), weight=1)
    root.columnconfigure((0,1,2,3),weight=1)
    return root
```

```

def createconnection() : #Create Connection to sqlite3
    global conn,cursor
    conn = sqlite3.connect('database/cryptobase.db')
    cursor = conn.cursor()

def loginclick() : #Login info Check
    global result, status_login
    global user
    global btc_coin, eth_coin, doge_coin, kub_coin, bnb_coin
    #user = userentry.get()
    if userinfo.get() == "" :
        messagebox.showwarning("Cryptonite : Warning","Please enter
your username.")
        userentry.focus_force()
    else:
        if pwdinfo.get() == "" :
            messagebox.showwarning("Cryptonite : Warning","Please enter
your password.")
            pwdentry.focus_force()
        else:
            sql = "SELECT * FROM user_data WHERE username=? and
password=?"
            cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
            result = cursor.fetchone()
            if result :
                messagebox.showinfo("Cryptonite : Successfully","Login
Successfully.")
                user = userentry.get()
                status_login = True
                #Import API
                response = requests.get(API_HOST +
"/api/market/ticker")
                result_coin = response.json()
                #BTC
                btc_sym = "THB_BTC"
                btc_data = result_coin[btc_sym]
                btc_coin = btc_data["last"]
                #ETH
                eth_sym = 'THB_ETH'
                eth_data = result_coin[eth_sym]
                eth_coin = eth_data["last"]
                #DOGE
                doge_sym = 'THB_DOGE'
                doge_data = result_coin[doge_sym]
                doge_coin = doge_data["last"]

```

```

#KUB
kub_sym = 'THB_KUB'
kub_data = result_coin[kub_sym]
kub_coin = kub_data["last"]
#BNB
bnb_sym = 'THB_BNB'
bnb_data = result_coin[bnb_sym]
bnb_coin = bnb_data["last"]
profile_fn()
else:
    messagebox.showwarning("Cryptonite :"
Warning","Incorrect Username or password")
    userentry.delete(0,END)
    pwdentry.delete(0,END)
    userentry.focus_force()

def registerlogin_fn():
    global userentry, pwdentry, reglog_f

    reglog_frm = Frame(root, bg='#EBFCE0')
    reglog_frm.place(x=0, y=0, width=w, height=h)
    reglog_frm.rowconfigure((0,1,2,3), weight=1)
    reglog_frm.columnconfigure((0,1,2,3), weight=1)
    bg = Label(reglog_frm, image=lobby_bg)
    bg.place(x=0,y=0,width=w,height=h)
    reglog_f = Frame(reglog_frm, bg="#C8D9DD", bd=10)
    reglog_f.rowconfigure((0,1,2,3,4), weight=1)
    reglog_f.columnconfigure((0,1),weight=1)
    reglog_f.grid(row=1,column=1, columnspan=2, rowspan=2
,sticky="news")

    Label(reglog_f,image=main_img,bg='#C8D9DD',
bd=0).grid(row=0,columnspan=2)
    Label(reglog_f, text="Account Login", font="Calibri 16 bold",
bg='#C8D9DD',fg='#414444', bd=10).grid(row=1, columnspan=2)
    Label(reglog_f,text='Username :',bg='#C8D9DD', fg='#404C50',
width=15).grid(row=2,column=0, sticky=E)
    userentry = Entry(reglog_f, bg='#D6E9E1', textvariable=userinfo)
    userentry.grid(row=2,column=1,sticky='w',padx=15)
    Label(reglog_f, text='Password :',bg='#C8D9DD', fg='#404C50',
width=15).grid(row=3,column=0,sticky=E)
    pwdentry = Entry(reglog_f, bg='#D6E9E1', textvariable=pwdinfo,
show='*')
    pwdentry.grid(row=3,column=1,sticky='w',padx=15)

```

```

        Button(reglog_f, image=btn_regis,bg ='#C8D9DD',command=regis_fn,
bd=0).grid(row=4, column=0, ipady=15 , sticky=E)
        Button(reglog_f, image=btn_login, command=loginclick, bd=0,
bg= '#C8D9DD').grid(row=4, column=1, ipady=15 , sticky=W,padx=70)

        userentry.focus_force()

def regis_fn():
    global firstname, lastname, phone_number, username, password, cfpwd
    global regis_f, reg_right, status_regis
    status_regis = True
    regis_f = Frame(root, bg='pink')
    regis_f.place(x=0,y=0,width=w,height=h)
    bg = Label(regis_f, image=login_bg)
    bg.place(x=0,y=0,width=w,height=h)
    regis_f.rowconfigure((0,1,2), weight=1)
    regis_f.columnconfigure((0,1), weight=1)
    #Top
    reg_top = Frame(regis_f, bg='white')
    reg_top.grid(row=0, columnspan=3, sticky='news')
    Label(reg_top, image=main_img, bg='white').grid(row=0, column=0)
    Label(reg_top, text='REGISTER Account', font="Impact 30 bold
underline", bg='white').grid(row=0, column=1, sticky='news')
    #Left
    reg_left = LabelFrame(regis_f, text='Register Account')
    reg_left.rowconfigure((0,1,2,3,4,5), weight=1)
    reg_left.columnconfigure((0,1), weight=1)
    reg_left.grid(column=0, row=1, sticky='news', ipadx=300, ipady=100)
    reg_left.option_add("*font", "Verdana 16")
    Label(reg_left, text='First name :').grid(row=0, column=0,
sticky=W)
    firstname = Entry(reg_left, width=15, textvariable=fname)
    firstname.grid(row=0, column=0)
    Label(reg_left, text='Last name :').grid(row=1, column=0, sticky=W)
    lastname = Entry(reg_left, width=15, textvariable=lname)
    lastname.grid(row=1, column=0)
    Label(reg_left, text='Phone \nnumber :').grid(row=2, column=0,
sticky=W)
    phone_number = Entry(reg_left, width=15, textvariable=phonenum)
    phone_number.grid(row=2, column=0)
    Label(reg_left, text='Username :').grid(row=3, column=0, sticky=W)
    username = Entry(reg_left, width=15, textvariable=user_name)
    username.grid(row=3, column=0)
    Label(reg_left, text='Password :').grid(row=0, column=1, sticky=W)
    password = Entry(reg_left, width=15, textvariable=pwdinfo)
    password.grid(row=0, column=1)

```

```

    Label(reg_left, text='Confirm \nPassword :').grid(row=1, column=1,
sticky=W)
    cfpwd = Entry(reg_left, width=15, textvariable=cfpwdinfo, show='*')
    cfpwd.grid(row=1, column=1)
    Button(reg_left, image=btn_regis, bd=0,
command=registration).grid(row=2, column=1, sticky=W)
    #Right
    reg_right = LabelFrame(regis_f, bg='#C4DDFF', text='Avatar')
    reg_right.rowconfigure((0,1), weight=1)
    reg_right.columnconfigure((0,1,2), weight=1)
    reg_right.grid(column=1, row=1, sticky='news', ipadx=120)
    Radiobutton(reg_right, text='1.', bg='#C4DDFF',
variable=avatarinfo, value='1', command=avatarselect).grid(row=0,
column=0, sticky=W)
    Radiobutton(reg_right, text='2.', bg='#C4DDFF',
variable=avatarinfo, value='2', command=avatarselect).grid(row=0,
column=1, sticky=W)
    Radiobutton(reg_right, text='3.', bg='#C4DDFF',
variable=avatarinfo, value='3', command=avatarselect).grid(row=0,
column=2, sticky=W)
    #Bottom
    Button(regis_f, image=ico_return, compound=LEFT, text='Back',
bg='#8FBDD3', command=backtologin).grid(row=2, column=0 ,sticky=W,
padx=5)

    firstname.focus_force()

def avatarselect():
    global avatar
    avatar = avatarinfo.get()
    if avatar == '1' :
        Label(reg_right, image=doge_pf, compound=TOP,
text='Dogey').grid(row=1, columnspan=3)
    elif avatar == '2' :
        Label(reg_right, image=mr_pf, compound=TOP,
text='Mr.increed').grid(row=1, columnspan=3)
    elif avatar == '3' :
        Label(reg_right, image=alien_pf, compound=TOP,
text='Alien').grid(row=1, columnspan=3)

def registration():
    global btc, eth, doge, kub, bnb, nft1, nft2, nft3, nft4, nft5,
nft6,
near

    if fname.get() == '' :

```

```

        messagebox.showwarning("Cryptonite : Warning",'Enter First name
First')
        firstname.focus_force()
    elif lname.get() == '' :
        messagebox.showwarning("Cryptonite : Warning",'Enter Last name
first')
        lastname.focus_force()
    elif phonenum.get() == '':
        messagebox.showwarning("Cryptonite : Warning",'Enter Phone
number first')
        phone_number.focus_force()
    elif phonenum.get().isnumeric() == False :
        messagebox.showwarning("Cryptonite : Warning",'Please check
your Phone number')
        phone_number.focus_force()
    elif len(phonenum.get()) != 10 :
        messagebox.showwarning("Cryptonite : Warning",'Enter 10 digits
of your Phone number')
        phone_number.focus_force()
    elif user_name.get() == 0 :
        messagebox.showwarning("Cryptonite : Warning",'Enter User name
first')
        username.focus_force()
    elif pwdinfo.get() == '' :
        messagebox.showwarning("Cryptonite : Warning",'Enter Password
first')
        password.focus_force()
    elif cfpwdinfo.get() == '' :
        messagebox.showwarning("Cryptonite : Warning",'Confirm Password
first')
        cfpwd.focus_force()
    elif avatarinfo.get() == '' :
        messagebox.showwarning("Cryptonite : Warning",'Please choose
Avatar first')
    else :
        idgen = str(ord(fname.get()[0])) + str(ord(lname.get()[1])) +
str(ord(lname.get()[0])) + str(ord(lname.get()[1])) + phonenum.get()[6]
+ phonenum.get()[7] + phonenum.get()[8] + phonenum.get()[9]
        id = int(idgen)
        #starter pack
        money_thb = 500
        btc = 0
        eth = 0
        doge = 0
        kub = 0
        bnb = 0

```

```

nft1 = 0
nft2 = 0
nft3 = 0
nft4 = 0
nft5 = 0
nft6 = 0
near = 0
messagebox.showinfo("Cryptonite : Your ID", 'Please Keep this
ID %s' %(idgen))
sql_chk = 'SELECT * from user_data WHERE id=?'
cursor.execute(sql_chk, [idgen])
result_chk = cursor.fetchall()
if result_chk:
    messagebox.showwarning("Cryptonite : ", 'This Registration
information is already exist\nPlease try again')
    firstname.select_range(0,END)
    lastname.select_range(0,END)
    phone_number.select_range(0,END)
    username.select_range(0,END)
    password.select_range(0,END)
    cfpwd.select_range(0,END)
    firstname.focus_force()
else:
    #check password and cf password
    if pwdinfo.get() == cfpwdinfo.get():
        sql = '''INSERT INTO user_data
(id,name,surname,username,password,phonenum,image_avatar,money,btc,e
th,doge,kub,bnb,nft1,nft2,nft3,nft4,nft5,nft6,near) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)'''
        cursor.execute(sql, [id, fname.get(), lname.get(),
user_name.get(), pwdinfo.get(), phonenum.get(), avatarinfo.get(),
money_thb, btc, eth, doge, kub, bnb, nft1, nft2, nft3, nft4, nft5,
nft6, near])
        conn.commit()
        retrivedata()
        messagebox.showinfo("Cryptonite :
Successfully", 'Registration Successfully')
        firstname.delete(0,END)
        lastname.delete(0,END)
        phone_number.delete(0,END)
        username.delete(0,END)
        password.delete(0,END)
        cfpwd.delete(0,END)
    else:
        messagebox.showwarning("Cryptonite : Confirm password
", 'Confirm password is not matched')

```

```

def profileClick():
    if userentry.get() == '' or pwdentry.get() == '' :
        messagebox.showwarning('Cryptonite : You must login first.',
'You are not logged in, You must login first')
        backtologin()
    else :
        profile_fn()

def profile_fn():
    global mycoin_ent, fullname, mycoin, result_user
    fullname = result[1] + ' ' + result[2]
    root.title("Cryptonite : Welcome "+ fullname)
    sql_user = "SELECT * FROM user_data WHERE username=?"
    cursor.execute(sql_user , [user])
    result_user = cursor.fetchone()
    print(result_user)
    if status_login == False :
        messagebox.showwarning('Cryptonite : You must login first.',
'You are not logged in, You must login first')
        backtologin()
    else:
        global pf_f
        #profile_fn Main Frame
        pf_f = Frame(root, bg="lightyellow")
        bg = Label(pf_f, image=login_bg)
        bg.place(x=0,y=0,width=w,height=h)
        pf_f.place(x=0, y=0, width=w, height=h)
        pf_f.rowconfigure((0,1,2), weight=1)
        pf_f.columnconfigure((0,1), weight=1)
        pf_f.option_add("*font", "Verdana 16")
        #profile_fn Top Frame
        pf_f_top = Frame(pf_f, bg='white')
        pf_f_top.grid(row=0, columnspan=3, sticky='news')
        pf_f_top.columnconfigure((0,1,2,3,4), weight=1)
        if result_user:
            if result_user[6] == 1 :
                Label(pf_f_top, image=doge_pf, bg='white').grid(row=0,
column=0, sticky='news')
            elif result_user[6] == 2 :
                Label(pf_f_top, image=mr_pf, bg='white').grid(row=0,
column=0, sticky='news')
            elif result_user[6] == 3 :
                Label(pf_f_top, image=alien_pf, bg='white').grid(row=0,
column=0, sticky='news')

```

```

        Label(pf_f_top, text= fullname, font="Impact 20 bold ",
bg='white').grid(row=0, column=1, sticky='sw')
        Button(pf_f_top, image=btn_addcredit, bg="white", bd=0,
command=addcredit_fn).grid(row=0, column=2, sticky=S,padx=20)
        Button(pf_f_top, image=btn_marketcap, bg="white", bd=0,
command=margetcap_fn).grid(row=0, column=3, sticky=S)
        Button(pf_f_top, image=btn_totalppt, bg="white", bd=0,
command=totalproperty_fn).grid(row=0, column=4, sticky=S, padx=20)
    #profile_fn Left Frame
    mycoin = []
    if result[8] >= 0 :
        mycoin.append('BTC')
    if result[9] >= 0 :
        mycoin.append('ETH')
    if result[10] >= 0 :
        mycoin.append('DOGE')
    if result[11] >= 0 :
        mycoin.append('KUB')
    if result[12] >= 0 :
        mycoin.append('BNB')
    else :
        messagebox.showinfo('Cryptonite : Warnning', 'You do not
have coin')
    pf_f_left = LabelFrame(pf_f, text='Currency')
    pf_f_left.rowconfigure((0,1), weight=1)
    pf_f_left.rowconfigure(2, weight=2)
    pf_f_left.columnconfigure((0,1), weight=1)
    pf_f_left.grid(row=1, column=0, sticky='news', ipady=100)
    Label(pf_f_left, text="THB", fg="#355037",font='Verdana 20
bold').grid(row=0, column=0, sticky=W ,padx=10)
    total_thb = Entry(pf_f_left, width=21, font='Verdana 14',
justify=RIGHT)
    total_thb.grid(row=0, column=0, sticky=E)
    mycoin_op = OptionMenu(pf_f_left, mycoinopinfo, *mycoin)
    mycoin_op.grid(row=1, column=0, sticky=W, padx=10)
    mycoin_ent = Entry(pf_f_left, width=19, justify=RIGHT)
    mycoin_ent.grid(row=1, column=0, sticky=E)
    Button(pf_f_left, image=ico_search, command=searchcoin,
bd=0).grid(row=1, column=1, sticky=W)
    Button(pf_f_left, image=btn_getcoin, command=getcoin_fn,
bd=0).grid(row=2, column=0)
    Button(pf_f_left, image=btn_sellcoin, command=sellcoin_fn,
bd=0).grid(row=2, column=0, sticky=E)
    #profile_fn Right Frame
    pf_f_right = LabelFrame(pf_f, bg='#D1D1D1', text='NFT')
    pf_f_right.option_add('*font', 'Verdana 16')

```

```

        pf_f_right.rowconfigure((0,1,2), weight=1)
        pf_f_right.columnconfigure((0,1,2), weight=1)
        pf_f_right.grid(row=1, column=1, sticky="news")
        Button(pf_f_right, image=btn_seenft, bg='#D1D1D1', bd=0,
command=seeNFT).grid(row=0, column=0)
        Button(pf_f_right, image=btn_getnft, bg='#D1D1D1', bd=0,
command=getnft_fn).grid(row=0, column=1)
        Button(pf_f_right, image=btn_sellnft, bg='#D1D1D1', bd=0,
command=sellnft_fn).grid(row=0, column=2)
        #Insert Zone
        mycoinopinfo.set('My Coin')
        total_thb.insert(0, ('%0.2f'%(result[7])))

def seeNFT():
    global seenft_f,seenft_fleft,seenft_fright

    seenft_f = Frame(root,bg='blue')
    seenft_f.rowconfigure((0,1,2,3),weight=1)
    seenft_f.columnconfigure((0,1,2,3),weight=1)
    seenft_f.place(x=0,y=0,width=w,height=h)
    bg = Label(seenft_f, image=login_bg)
    bg.place(x=0,y=0,width=w,height=h)

    seenft_fleft = LabelFrame(seenft_f,text='My Profile')
    seenft_fleft.rowconfigure((0,1,2,3,4,5,6,7,8,9),weight=1)
    seenft_fleft.columnconfigure((0,1),weight=1)
    seenft_fleft.grid(row=0,column=0,rowspan=4,columnspan=2,sticky='news')

    seenft_fright = LabelFrame(seenft_f,text='NFT')
    seenft_fright.rowconfigure((0,1,2,3),weight=1)
    seenft_fright.columnconfigure((0,1,2),weight=1)
    seenft_fright.grid(row=0,column=2,rowspan=5,columnspan=3,sticky='news', ipadx=200)

    #ล้วนชื่อ
    if result_user:
        if result_user[6] == 1 :
            Label(seenft_fleft, image=doge_pf).grid(row=0, column=0,
columnspan=2, sticky='news')
        elif result_user[6] == 2 :
            Label(seenft_fleft, image=mr_pf).grid(row=0, column=0,
columnspan=2, sticky='news')
        elif result_user[6] == 3 :
            Label(seenft_fleft, image=alien_pf).grid(row=0, column=0,
columnspan=2, sticky='news')

```

```

    Label(seenft_fleft, text= fullname, font="Impact 20 bold
").grid(row=1, column=0, columnspan=2)

    Button(seenft_f, text='Back To Profile', image=ico_return,
compound=LEFT, command=profile_fn).grid(row=4, column=0,
sticky=W, ipadx=20, ipady=10)

#ส่วนขวา
if result[13] == 0 :
    Label(seenft_fright,image=img.blurmonalisa).grid(row=0,column=0
,sticky='news')
else :
    Label(seenft_fright,image=img.mona, compound=TOP,
text='Monalisa', font="Impact 15 bold
").grid(row=0,column=0,sticky='news')
    if result[14] == 0 :
        Label(seenft_fright,image=img.blur3dboy).grid(row=0,column=1,st
icky='news')
    else :
        Label(seenft_fright,image=img.nft3D, compound=TOP,
text='3DBoy', font="Impact 15 bold
").grid(row=0,column=1,sticky='news')
    if result[15] == 0 :
        Label(seenft_fright,image=img.blurheathead).grid(row=0,column=2
,sticky='news')
    else :
        Label(seenft_fright,image=img.heath, compound=TOP,
text='Heathead', font="Impact 15 bold
").grid(row=0,column=2,sticky='news')
    if result[16] == 0 :
        Label(seenft_fright,image=img.bluranti).grid(row=1,column=0,st
icky='news')
    else :
        Label(seenft_fright,image=img.anticoding, compound=TOP,
text='Anticoding',font="Impact 15 bold
").grid(row=1,column=0,sticky='news')
    if result[17] == 0 :
        Label(seenft_fright,image=img.blurboredm).grid(row=1,column=1,s
ticker='news')
    else :
        Label(seenft_fright,image=img.bored, compound=TOP,
text='BoredM', font="Impact 15 bold
").grid(row=1,column=1,sticky='news')
    if result[18] == 0 :
        Label(seenft_fright,image=img.blurlotus).grid(row=1,column=2,st
icky='news')

```

```

    else :
        Label(seenft_fright,image=img_lotus, compound=TOP, text='Lotus',
font="Impact 15 bold ").grid(row=1,column=2,sticky='news')

def searchcoin():
    mycoin_ent.delete(0,END)
    if mycoinopinfo.get() == 'BTC':
        mycoin_ent.insert(0,('%.4f'%(result[8])))
    elif mycoinopinfo.get() == 'ETH':
        mycoin_ent.insert(0,('%.4f'%(result[9])))
    elif mycoinopinfo.get() == 'DOGE':
        mycoin_ent.insert(0,('%.4f'%(result[10])))
    elif mycoinopinfo.get() == 'KUB':
        mycoin_ent.insert(0,('%.4f'%(result[11])))
    elif mycoinopinfo.get() == 'BNB':
        mycoin_ent.insert(0,('%.4f'%(result[12])))

def margetcap_fn():
    global status_login
    if status_login == False :
        messagebox.showwarning('Cryptonite : You must login first.',
'You are not logged in, You must login first')
        backtologin()
    else:
        status_login = True
        global mkcap_f
        mkcap_f = Frame(root)
        mkcap_f.rowconfigure((0,1,2,3,4,5), weight=1)
        mkcap_f.columnconfigure((0,1,2,3), weight=1)
        mkcap_f.place(x=0, y=0, width=w, height=h)

        mkcap_ftop = Frame(mkcap_f, bg="white")
        mkcap_ftop.rowconfigure((0,1,2,3),weight=1)
        mkcap_ftop.columnconfigure((0,1,2,3,4,5),weight=1)
        mkcap_ftop.grid(row=0,column=0,columnspan=4,sticky='news')

        mkcap_fcenter = Frame(mkcap_f, bg="#D1D1D1")
        mkcap_fcenter.rowconfigure((0,1,2,3,4,5,6,7,8),weight=1)
        mkcap_fcenter.columnconfigure((0,1,2,3,4,5,6),weight=1)
        mkcap_fcenter.grid(row=1,column=0, rowspan=5, columnspan=6,sticky
='news')

        mkcap_fbotttom = Frame(mkcap_f,bg='pink')
        mkcap_fbotttom.rowconfigure((0,1,2,3),weight=1)

```

```

    mkap_fbottom.columnconfigure((0,1,2),weight=1)
    bg = Label(mkap_fbottom,image=login_bg)
    bg.place(x=0,y=0,width=w,height=h)
    mkap_fbottom.grid(row=6,column=0,columnspan=4,sticky='news')

    #Show market cap
    Label(mkap_ftop,text='Market Cap',bg='white',font='Verdana 20 bold').grid(row=1,column=0,sticky='w', padx=10)

        Label(mkap_ftop,text='Total',bg='white',font='Verdana 15 bold').grid(row=1,column=1,sticky='w')
        Label(mkap_ftop,text='%.2f'%(result[7]),bg='white',font='Verdana 15 bold').grid(row=1,column=2,sticky='w')
        Label(mkap_ftop,text='THB',bg='white',font='Verdana 15 bold').grid(row=1,column=3,sticky='w')

        Button(mkap_ftop, image=btn_getcoin, bd=0, bg='white', command=getcoin_fn).grid(row=1,column=4,ipadx=20,ipady=10)
        Button(mkap_ftop, image=btn_sellcoin, bg='white', bd=0, command=sellcoin_fn).grid(row=1, column=5)

        #data market cap
        Label(mkap_fcenter,text='BTC/THB',bg='#D1D1D1',font='Verdana 15 bold').grid(row=0,column=0,sticky='e')
        entry_btc = Entry(mkap_fcenter,width=20, justify=RIGHT)
        entry_btc.grid(row=0,column=1,sticky='w')
        Label(mkap_fcenter,image=img_bitcoin,bg='#D1D1D1').grid(row=1, column=1,sticky='w')

        Label(mkap_fcenter,text='DOGE/THB',bg='#D1D1D1',font='Verdana 15 bold').grid(row=0,column=2,sticky=E)
        entry_doge = Entry(mkap_fcenter,width=20, justify=RIGHT)
        entry_doge.grid(row=0,column=3,sticky='w')
        Label(mkap_fcenter,image=img_doge,bg='#D1D1D1').grid(row=1, column=3,sticky='w')

        Label(mkap_fcenter,text='ETH/THB',bg='#D1D1D1',font='Verdana 15 bold').grid(row=3,column=0,sticky='e')
        entry_eth = Entry(mkap_fcenter,width=20, justify=RIGHT)
        entry_eth.grid(row=3,column=1,sticky='w')
        Label(mkap_fcenter,image=img_eth,bg='#D1D1D1').grid(row=4, column=1,sticky='w')

        Label(mkap_fcenter,text='KUB/THB',bg='#D1D1D1',font='Verdana 15 bold').grid(row=3,column=2,sticky=E)
        entry_kub = Entry(mkap_fcenter,width=20, justify=RIGHT)

```

```

        entry_kub.grid(row=3,column=3,sticky='w')
        Label(mkcap_fcenter,image=img_bitkub,bg='#D1D1D1').grid(row=4,column=3,sticky='w')

        Label(mkcap_fcenter,text='BNB/THB',bg='#D1D1D1',font='Verdana
15 bold').grid(row=5,column=1,sticky='e')
        entry_bnb = Entry(mkcap_fcenter,width=20, justify=RIGHT)
        entry_bnb.grid(row=5,column=2,sticky='w')
        Label(mkcap_fcenter,image=img_bnb,bg='#D1D1D1').grid(row=6,colu
mnspace=3,sticky='e')
        #Insert Coin in Entry
        entry_btc.insert(0, btc_coin)
        entry_eth.insert(0, eth_coin)
        entry_doge.insert(0, doge_coin)
        entry_kub.insert(0, kub_coin)
        entry_bnb.insert(0, bnb_coin)
        #จำนวนเงิน
        Button(mkcap_fbbottom,text='Back To Profile', image=ico_return,
compound=LEFT,
command=profile_fn).grid(row=3,column=0,ipadx=20,ipady=10,sticky='w')
        Button(mkcap_fbbottom,text='Refresh', image=ico_reload,
compound=RIGHT,
command=margetcap_fn).grid(row=3,column=4,ipadx=20,ipady=10,sticky='w')
def addcredit_fn():
    global status_login, addcre_left
    if status_login == False :
        messagebox.showwarning('Cryptonite : You must login first.',
'You are not logged in, You must login first')
        backtologin()
    else:
        root.title('Cryptonite : Add Credit ID'+ ' '+ str(result[0]))
        status_login = True
        global addcre_f
        addcre_f = Frame(root, bg="lightgrey")
        addcre_f.rowconfigure((0,1,2), weight=1)
        addcre_f.columnconfigure((0), weight=1)
        addcre_f.place(x=0, y=0, width=w, height=h)
        addcre_f.option_add("*font", "Verdana 16")
        bg = Label(addcre_f, image=login_bg)
        bg.place(x=0,y=0,width=w,height=h)
        #Top
        addcre_top = Frame(addcre_f, bg='white')
        addcre_top.columnconfigure((0,1,2,3), weight=1)
        addcre_top.grid(row=0, column=0, sticky='news')
        Label(addcre_top, image=main_img, bg='white').grid(row=0,
column=0)

```

```

        Label(addcre_top, text="THB", fg="#355037", font='Verdana 20 bold', bg='white').grid(row=0, column=1, sticky=W)
        total_thb = Entry(addcre_top, width=20, font='Impact 30 bold', justify=RIGHT, bg='#D4F1F5')
        total_thb.grid(row=0, column=1)
        total_thb.insert(0, ('%.2f'%(result[7])))
        Button(addcre_top, image=ico_reload, bg='#E6FFE7', command=addcredit_fn).grid(row=0, column=2, sticky=W)
        Button(addcre_top, image=btn_getcoin, command=getcoin_fn, bd=0, bg='white').grid(row=0, column=3, sticky=W)
        #Left
        addcre_left = LabelFrame(addcre_f, text='Amount to add')
        addcre_left.rowconfigure((0,1,2,3), weight=2)
        addcre_left.columnconfigure((0,1), weight=1)
        addcre_left.grid(row=1, column=0, sticky='news', ipady=100)
        Radiobutton(addcre_left, text='500', font='Impact 20 bold', variable=amt_addcredit, value='1').grid(row=0, column=0, sticky=W, padx=10)
        Radiobutton(addcre_left, text='1000', font='Impact 20 bold', variable=amt_addcredit, value='2').grid(row=1, column=0, sticky=W, padx=10)
        Radiobutton(addcre_left, text='3000', font='Impact 20 bold', variable=amt_addcredit, value='3').grid(row=2, column=0, sticky=W, padx=10)
        Radiobutton(addcre_left, text='Enter amout', font='Verdana 16 ', variable=amt_addcredit, value='4', command=amt_fill).grid(row=0, column=1, sticky=W)
        Button(addcre_left, image=btn_confirmmpurches, command=confirm_addcredit, bd=0).grid(row=2, column=1, sticky=W)
        Button(addcre_f, image=ico_return, compound=LEFT, text='Back to Profile', command=profile_fn).grid(row=2, column=0 ,sticky=W, ipadx=20, ipady=15)

def amt_fill():
    global amt_ent
    amt_ent = Entry(addcre_left, width=15, font='Verdana 14', justify=RIGHT, textvariable=amt_entinfo)
    amt_ent.grid(row=1, column=1, sticky=W)

def confirm_addcredit():
    global result
    select_user = result[0]
    radio_add = amt_addcredit.get()
    money = 0
    if radio_add == '' :

```

```

        messagebox.showwarning('Cryptonite : Warnning', 'Please Select
Amout.')
    elif radio_add == '1' :
        purchasing_status = messagebox.askquestion('Cryptonite : 500
THB', 'Are you sure to purchase 500 THB.')
        if purchasing_status == 'yes' :
            money = result[7] + 500
            sql = '''
                UPDATE user_data
                SET money=?
                WHERE id=?
            '''
            ...
            cursor.execute(sql, [money, select_user])
            conn.commit()
            update = messagebox.showinfo('Cryptonite : Purchasing
Complete', 'Your 500 THB Purchasing Complete.')
            if update :
                sql = "SELECT * FROM user_data WHERE username=? and
password=?"
                cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                result = cursor.fetchone()
                addcredit_fn()
            else:
                sql = "SELECT * FROM user_data WHERE username=? and
password=?"
                cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                result = cursor.fetchone()
                addcredit_fn()
        elif radio_add == '2' :
            purchasing_status = messagebox.askquestion('Cryptonite : 1000
THB', 'Are you sure to purchase 1000 THB.')
            if purchasing_status == 'yes' :
                money = result[7] + 1000
                sql = '''
                    UPDATE user_data
                    SET money=?
                    WHERE id=?
                '''
                ...
                cursor.execute(sql, [money, select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite : Purchasing
Complete', 'Your 1000 THB Purchasing Complete.')
                if update :
                    sql = "SELECT * FROM user_data WHERE username=? and
password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.get()])

```

```

        result = cursor.fetchone()
        addcredit_fn()
    else:
        sql = "SELECT * FROM user_data WHERE username=? and
password=?"
        cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
        result = cursor.fetchone()
        addcredit_fn()
    elif radio_add == '3' :
        purchasing_status = messagebox.askquestion('Cryptonite : 500
THB', 'Are you sure to purchase 3000 THB.')
        if purchasing_status == 'yes' :
            money = result[7] + 3000
            sql = '''
                UPDATE user_data
                SET money=?
                WHERE id=?
            '''
            ...
            cursor.execute(sql, [money, select_user])
            conn.commit()
            update = messagebox.showinfo('Cryptonite : Purchasing
Complete', 'Your 3000 THB Purchasing Complete.')
            if update :
                sql = "SELECT * FROM user_data WHERE username=? and
password=?"
                cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                result = cursor.fetchone()
                addcredit_fn()
            else:
                sql = "SELECT * FROM user_data WHERE username=? and
password=?"
                cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                result = cursor.fetchone()
                addcredit_fn()
    elif radio_add == '4' :
        if int(amt_entinfo.get()) <= 0 :
            messagebox.showwarning('Cryptonite : Warnning', 'You need
purchase a minimum of 1 baht.')
            amt_ent.delete(0, END)
            amt_ent.focus_force()
        else:
            purchasing_status = messagebox.askquestion('Cryptonite :
%.2f THB'%(int(amt_entinfo.get())), 'Are you sure to purchase %.2f
THB.'%(int(amt_entinfo.get())))
            if purchasing_status == 'yes' :
                money = result[7] + int(amt_entinfo.get())

```

```

        sql = '''
                UPDATE user_data
                SET money=?
                WHERE id=?
            '''

        cursor.execute(sql, [money, select_user])
        conn.commit()
        update = messagebox.showinfo('Cryptonite : Purchasing
Complete', 'Your %0.2f THB Purchasing
Complete.'%int(amt_entinfo.get())))
        if update :
            sql = "SELECT * FROM user_data WHERE username=? and
password=?"
            cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
            result = cursor.fetchone()
            addcredit_fn()
        else:
            sql = "SELECT * FROM user_data WHERE username=? and
password=?"
            cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
            result = cursor.fetchone()
            addcredit_fn()

def getcoin_fn():
    global status_login, mycoin_ent2, getc_main, spin
    if status_login == False :
        messagebox.showwarning('Cryptonite : You must login first.',
'You are not logged in, You must login first')
        backtologin()
    else:
        root.title('Cryptonite : Get Coin %s$'%( ' *150, fullname))
        status_login = True
        global getc_f
        getc_f = Frame(root, bg="green")
        getc_f.rowconfigure((0,1,2,3), weight=1)
        getc_f.columnconfigure((0,1,2,3), weight=1)
        getc_f.place(x=0, y=0, width=w, height=h)
        bg = Label(getc_f, image=login_bg)
        bg.place(x=0,y=0,width=w,height=h)

        #Top
        getc_top = Frame(getc_f, bg='white')
        getc_top.columnconfigure((0,1,2,3), weight=1)
        getc_top.grid(row=0, columnspan=4, sticky='news')
        if result:
            if result[6] == 1 :

```

```

        Label(getc_top, image=doge_pf,
bg='white').grid(row=0, column=0, sticky='news')
    elif result[6] == 2 :
        Label(getc_top, image=mr_pf,
bg='white').grid(row=0, column=0, sticky='news')
    elif result[6] == 3 :
        Label(getc_top, image=alien_pf,
bg='white').grid(row=0, column=0, sticky='news')
    Label(getc_top, text="THB", fg="#355037",font='Verdana 20 bold',
bg='white').grid(row=0, column=1, sticky=W)
    total_thb = Entry(getc_top, width=20, font='Impact 30 bold',
justify=RIGHT, bg='#D4F1F5')
    total_thb.grid(row=0, column=1)
    total_thb.insert(0, ('%.2f'%(result[7])))
    Button(getc_top, image=btn_addcredit, bg='white', bd=0,
command=addcredit_fn).grid(row=0, column=2, sticky=E)
#Getcoin main
coin = ['BTC', 'ETH', 'DOGE', 'KUB', 'BNB']
getc_main = Frame(getc_f)
getc_main.rowconfigure((0,1,2,3), weight=1)
getc_main.columnconfigure((0,1,2), weight=1)
getc_main.grid(row=1, columnspan=4, sticky='news', ipady=100)
mycoin_op = OptionMenu(getc_main, mycoinopinfo, *coin)
mycoin_op.grid(row=1, column=0, padx=10, sticky=W)
mycoin_ent2 = Entry(getc_main, width=19, justify=RIGHT)
mycoin_ent2.grid(row=1, column=0, sticky=E)
mycoinopinfo.set('Get Coin')
Button(getc_main, image=ico_search, bd=0,
command=searchcrypto).grid(row=1, column=1, sticky=W)
    Button(getc_main, image=btn_confirm_purchases, bd=0,
command=confirm_getcoin).grid(row=3, column=0, sticky=E)
    spin = Spinbox(getc_main, textvariable=spincoin_info, from_=1,
to=10, width=8, justify=RIGHT)
    spin.grid(row=2, column=0, sticky=E)
    Button(getc_f, text='Back To Profile', image=ico_return,
compound=LEFT,
command=profile_fn).grid(row=3, column=0, ipadx=20, sticky='w', ipady=10)

def searchcrypto() :
    mycoin_ent2.delete(0, END)
    get = 0
    if mycoinopinfo.get() == 'BTC' :
        get = btc_coin * spincoin_info.get()
        mycoin_ent2.insert(0, get)
        Label(getc_main, image=img_bitcoin).grid(row=2, column=0,
sticky=W)

```

```

        elif mycoinopinfo.get() == 'ETH' :
            get = eth_coin * spincoin_info.get()
            mycoin_ent2.insert(0, eth_coin)
            Label(getc_main, image=img_eth).grid(row=2, column=0, sticky=W)
        elif mycoinopinfo.get() == 'DOGE' :
            get = doge_coin * spincoin_info.get()
            mycoin_ent2.insert(0, doge_coin)
            Label(getc_main, image=img_doge).grid(row=2, column=0,
sticky=W)
        elif mycoinopinfo.get() == 'KUB' :
            get = kub_coin * spincoin_info.get()
            mycoin_ent2.insert(0, kub_coin)
            Label(getc_main, image=img_bitkub).grid(row=2, column=0,
sticky=W)
        elif mycoinopinfo.get() == 'BNB' :
            get = bnb_coin * spincoin_info.get()
            mycoin_ent2.insert(0, bnb_coin)
            Label(getc_main, image=img_bnb).grid(row=2, column=0, sticky=W)

def confirm_getcoin() :
    global result
    status = messagebox.askquestion('Cryptonite : Get Coin ', 'Are you
sure to Purchasing %d %s'%(spincoin_info.get(), mycoinopinfo.get()))
    if status == 'yes' :
        print(mycoinopinfo.get())
        if mycoinopinfo.get() == 'BTC' :
            payment = btc_coin * spincoin_info.get()
            if result[7] >= payment :
                status_payment = messagebox.askquestion('Cryptonite :
Confirm Purchasing', 'Confirm to paid %i'%(payment))
                if status_payment == 'yes':
                    select_user = result[0]
                    money = result[7] - payment
                    newcoin = result[8] + spincoin_info.get()
                    sql = '''
                        UPDATE user_data
                        SET money=?, btc=?
                        WHERE id=?
                    '''
                    cursor.execute(sql, [money, newcoin, select_user])
                    conn.commit()
                    update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Purchasing is Successfully.')
                    if update:
                        sql = "SELECT * FROM user_data WHERE username=?"
                        and password=?"

```

```

        cursor.execute(sql,[userinfo.get(),pwdinfo.get()
)])]
        result = cursor.fetchone()
        getcoin_fn()
        spin.delete(0, END)
        spin.insert(0,1)
        if status_payment == 'no' :
            messagebox.showinfo('Cryptonite : Get Coin', 'The
Payment was cancelled.')
        else :
            messagebox.showwarning('Cryptonite : Warnning', 'You do
not have enough credit\nyou need to add credit')
            status_addcredit = messagebox.askquestion('Cryptonite :
Add Credit', 'Do you want to Add Credit?')
            if status_addcredit == 'yes' :
                addcredit_fn()
            else:
                getcoin_fn()
                spin.delete(0, END)
                spin.insert(0,1)
        elif mycoinopinfo.get() == 'ETH' :
            payment = eth_coin * spincoin_info.get()
            if result[7] >= payment :
                status_payment = messagebox.askquestion('Cryptonite :
Confirm Purchasing', 'Confirm to paid %%(payment)')
                if status_payment == 'yes':
                    select_user = result[0]
                    money = result[7] - payment
                    newcoin = result[9] + spincoin_info.get()
                    sql = '''
                        UPDATE user_data
                        SET money=?, eth=?
                        WHERE id=?
                    '''
                    cursor.execute(sql, [money, newcoin, select_user])
                    conn.commit()
                    update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Purchasing is Successfully.')
                    if update:
                        sql = "SELECT * FROM user_data WHERE username=?"
                        and password=?""
                        cursor.execute(sql,[userinfo.get(),pwdinfo.get(
)])]
                        result = cursor.fetchone()
                        getcoin_fn()
                        spin.delete(0, END)

```

```

                spin.insert(0,1)
        if status_payment == 'no' :
            messagebox.showinfo('Cryptonite : Get Coin', 'The
Payment was cancelled.')
        else :
            messagebox.showwarning('Cryptonite : Warnning', 'You do
not have enough credit\nyou need to add credit')
            status_addcredit = messagebox.askquestion('Cryptonite :
Add Credit', 'Do you want to Add Credit?')
            if status_addcredit == 'yes' :
                addcredit_fn()
            else:
                getcoin_fn()
                spin.delete(0, END)
                spin.insert(0,1)
        elif mycoinopinfo.get() == 'DOGE' :
            payment = doge_coin * spincoin_info.get()
            if result[7] >= payment :
                status_payment = messagebox.askquestion('Cryptonite :
Confirm Purchasing', 'Confirm to paid %i'%(payment))
                if status_payment == 'yes':
                    select_user = result[0]
                    money = result[7] - payment
                    newcoin = result[10] + spincoin_info.get()
                    sql = '''
                        UPDATE user_data
                        SET money=?, doge=?
                        WHERE id=?
                    ...
                    cursor.execute(sql, [money, newcoin, select_user])
                    conn.commit()
                    update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Purchasing is Successfully.')
                    if update:
                        sql = "SELECT * FROM user_data WHERE username=?"
                        and password=?"
                        cursor.execute(sql,[userinfo.get(),pwdinfo.get(
                            )])
                        result = cursor.fetchone()
                        getcoin_fn()
                        spin.delete(0, END)
                        spin.insert(0,1)
                    if status_payment == 'no' :
                        messagebox.showinfo('Cryptonite : Get Coin', 'The
Payment was cancelled.')
                    else :

```

```

        messagebox.showwarning('Cryptonite : Warnning', 'You do
not have enough credit\nyou need to add credit')
        status_addcredit = messagebox.askquestion('Cryptonite :
Add Credit', 'Do you want to Add Credit?')
        if status_addcredit == 'yes' :
            addcredit_fn()
        else:
            getcoin_fn()
            spin.delete(0, END)
            spin.insert(0,1)
    elif mycoinopinfo.get() == 'KUB' :
        payment = kub_coin * spincoin_info.get()
        if result[7] >= payment :
            status_payment = messagebox.askquestion('Cryptonite :
Confirm Purchasing', 'Confirm to paid %i'%(payment))
            if status_payment == 'yes':
                select_user = result[0]
                money = result[7] - payment
                newcoin = result[11] + spincoin_info.get()
                sql = '''
                    UPDATE user_data
                    SET money=?, kub=?
                    WHERE id=?
                '''
                cursor.execute(sql, [money, newcoin, select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Purchasing is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE username=? and password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                    result = cursor.fetchone()
                    getcoin_fn()
                    spin.delete(0, END)
                    spin.insert(0,1)
                if status_payment == 'no' :
                    messagebox.showinfo('Cryptonite : Get Coin', 'The
Payment was cancelled.')
                else :
                    messagebox.showwarning('Cryptonite : Warnning', 'You do
not have enough credit\nyou need to add credit')
                    status_addcredit = messagebox.askquestion('Cryptonite :
Add Credit', 'Do you want to Add Credit?')
                    if status_addcredit == 'yes' :

```

```

        addcredit_fn()
    else:
        getcoin_fn()
        spin.delete(0, END)
        spin.insert(0,1)
    elif mycoinopinfo.get() == 'BNB' :
        payment = bnb_coin * spincoin_info.get()
        if result[7] >= payment :
            status_payment = messagebox.askquestion('Cryptonite : Confirm Purchasing', 'Confirm to paid %i'%(payment))
            if status_payment == 'yes':
                select_user = result[0]
                money = result[7] - payment
                newcoin = result[8] + spincoin_info.get()
                sql = '''
                    UPDATE user_data
                    SET money=?, bnb=?
                    WHERE id=?
                    ...
                '''
                cursor.execute(sql, [money, newcoin, select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite : Successfully', 'Your Purchasing is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE username=? and password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                    result = cursor.fetchone()
                    getcoin_fn()
                    spin.delete(0, END)
                    spin.insert(0,1)
                    if status_payment == 'no' :
                        messagebox.showinfo('Cryptonite : Get Coin', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Warnning', 'You do not have enough credit\nyou need to add credit')
                        status_addcredit = messagebox.askquestion('Cryptonite : Add Credit', 'Do you want to Add Credit?')
                        if status_addcredit == 'yes' :
                            addcredit_fn()
                        else:
                            getcoin_fn()
                            spin.delete(0, END)
                            spin.insert(0,1)

```

```

else:
    messagebox.showinfo('Cryptonite : Get Coin', 'The Purchasing
was cancelled.')

def getnft_fn():
    global near, nft_status
    nft_status = 0
    near = 352.09
    global status_login
    if status_login == False :
        messagebox.showwarning('Cryptonite : You must login first.',
'You are not logged in, You must login first')
        backtologin()
    else:
        status_login = True
        global getnft_f
        root.title('Cryptonite : Get NFT %s%s'%( ' '*151, fullname))
        getnft_f = Frame(root, bg="blue")
        getnft_f.rowconfigure((0,1,2,3,4), weight=1)
        getnft_f.columnconfigure((0,1,2,3,4,5), weight=1)
        getnft_f.place(x=0, y=0, width=w, height=h)

        global
getnft_ftop,getnft_fleft,getnft_fcenter,getnft_fright,getnft_fbottom
        getnft_ftop = Frame(getnft_f, bg="white")
        getnft_ftop.rowconfigure((0,1,2,3), weight=1)
        getnft_ftop.columnconfigure((0,1,2,3,4), weight=1)
        getnft_ftop.grid(row=0, column=0, columnspan=6, sticky='news')

        getnft_fleft = LabelFrame(getnft_f, text="Get Near")
        getnft_fleft.rowconfigure((0,1,2,3,4,5,6,7,8,9), weight=1)
        getnft_fleft.columnconfigure((0,1,2), weight=1)
        getnft_fleft.grid(row=1, column=0, rowspan=10, columnspan=2,
sticky='news')

        getnft_fcenter = Frame(getnft_f,bg='#D1D1D1')
        getnft_fcenter.rowconfigure((0,1,2,3,4),weight=1)
        getnft_fcenter.columnconfigure((0,1,2),weight=1)
        getnft_fcenter.grid(row=1,column=2,rowspan=5,columnspan=2,sticky='n
ews')

        getnft_fright = Frame(getnft_f, bg="#E8F0F2")
        getnft_fright.rowconfigure((0,1,2,3,4,5), weight=1)
        getnft_fright.columnconfigure((0,1), weight=1)
        getnft_fright.grid(row=1, column=4, rowspan=5, columnspan=2,
sticky='news')

```

```

getnft_fbottom = Frame(getnft_f, bg="lightgreen")
bg = Label(getnft_fbottom,image=login_bg)
bg.place(x=0,y=0,width=w,height=h)
getnft_fbottom.rowconfigure((0,1,2), weight=1)
getnft_fbottom.columnconfigure((0,1,2), weight=1)
getnft_fbottom.grid(row=5, column=0,columnspan=2, sticky='news')

#ล้วนบน
Label(getnft_ftop,text='Get NFT',bg='white',font='Vadana 20 bold').grid(row=1,column=0,sticky='nw')
Label(getnft_ftop,text='Total',bg='white',font='Verdana 20 bold').grid(row=1,column=1,sticky='w')
Label(getnft_ftop,text=( '%.2f' %(result[7])),bg='white',font='Verda na 20 bold').grid(row=1,column=2,sticky='w')
Label(getnft_ftop,text='THB',bg='white',font='Verdana 20 bold').grid(row=1,column=3,sticky='w')
Button(getnft_ftop,image=btn_addcredit,bg='white',bd=0,command=addcredit_fn).grid(row=1,column=4)

#ล้วนกลาง
if result[13] == 1 :
    Button(getnft_fcenter,image=img_soldmonalisa,text='Sold Out',compound=TOP,command=getNFT_monalisa).grid(row=0,column=0,sticky='w',ipadx=30,ipady=30)
else :
    Button(getnft_fcenter,image=img_mona,text='Monalisa',compound=TOP,command=getNFT_monalisa).grid(row=0,column=0,sticky='w',ipadx=30,ipady=30)
if result[14] == 1 :
    Button(getnft_fcenter,image=img_sold3d,text='Sold Out',compound=TOP,command=getNFT_3dboy).grid(row=0,column=1,sticky='w',ipadx=30,ipady=30)
else :
    Button(getnft_fcenter,image=img_nft3D,text='3DBoy',compound=TOP,command=getNFT_3dboy).grid(row=0,column=1,sticky='w',ipadx=30,ipady=30)
if result[15] == 1 :
    Button(getnft_fcenter,image=img_soldheat,text='Sold Out',compound=TOP,command=getNFT_heath).grid(row=0,column=2,sticky='w',ipadx=30,ipady=30)
else :
    Button(getnft_fcenter,image=img_heath,text='Heathead',compound=TOP,command=getNFT_heath).grid(row=0,column=2,sticky='w',ipadx=30,ipady=30)
if result[16] == 1 :

```

```

        Button(getnft_fcenter,image=img_soldanti,text='Sold
Out',compound=TOP,command=getNFT_anti).grid(row=1,column=0,sticky='w',
ipadx=30,ipady=30)
    else :
        Button(getnft_fcenter,image=img_anti,text='Anticoding',compound
=TOP,command=getNFT_anti).grid(row=1,column=0,sticky='w',ipadx=30,ipady
=30)
    if result[17] == 1 :
        Button(getnft_fcenter,image=img_soldlotus,text='Sold
Out',compound=TOP,command=getNFT_lotus).grid(row=1,column=1,sticky='w',
ipadx=30,ipady=30)
    else:
        Button(getnft_fcenter,image=img_lotus,text='Lotus',compound=TOP
,command=getNFT_lotus).grid(row=1,column=1,sticky='w',ipadx=30,ipady=30
)
    if result[18] == 1 :
        Button(getnft_fcenter,image=img_soldboredm,text='Sold
Out',compound=TOP,command=getNFT_boredm).grid(row=1,column=2,sticky='w'
,ipadx=30,ipady=30)
    else :
        Button(getnft_fcenter,image=img_bored,text='BoredM',compound=TO
P,command=getNFT_boredm).grid(row=1,column=2,sticky='w',ipadx=30,ipady=
30)

#Get Near
global getnear_ent, getnear_spn
Label(getnft_fleft, text='Near').grid(row=0, column=1, sticky=W)
mynear = Entry(getnft_fleft,width=10, justify=RIGHT)
mynear.grid(row=0, column=1, sticky=E)
mynear.insert(0, result[19])
Label(getnft_fleft, text='Amout :').grid(row=1, column=1, sticky=W)
Button(getnft_fleft, image=ico_search, bd=0,
command=calnear).grid(row=1,column=2,sticky=W)
getnear_spn = Spinbox(getnft_fleft,width=7,from_=0,to=20,
justify=RIGHT, textvariable=getnearspninfo)
getnear_spn.grid(row=1,column=1,sticky=E)

Label(getnft_fleft, text='THB :').grid(row=2, column=1, sticky=W)
getnear_ent = Entry(getnft_fleft,width=10, justify=RIGHT)
getnear_ent.grid(row=2,column=1,sticky=E)
Button(getnft_fleft,image=btn_getnear, bd=0,
command=getnear).grid(row=3,column=1,ipadx=5,ipady=5,sticky=E)

#Back Profile

```

```

        Button(getnft_fbottom, image=ico_return, compound=LEFT, text='Back
To Profile', command=profile_fn).grid(row=2, column=0, ipadx=10, ipady=10,
sticky=W)
def calnear() :
    getnear_ent.delete(0, END)
    totalnear = near * getnearspninfo.get()
    getnear_ent.insert(0, totalnear)

def getnear() :
    global result
    select_user = result[0]
    status = messagebox.askquestion('Cryptonite : Get Near ', 'Are you
sure to Get %i %s%(getnearspninfo.get(), 'near'))
    if status == 'yes' :
        print('Get Near')
        near_price = near * getnearspninfo.get()
        if result[7] >= near_price :
            status_near = messagebox.askquestion('Cryptonite : Confirm
Purchasing', 'Confirm to paid %i %(near_price)')
            if status_near == 'yes':
                select_user = result[0]
                money = result[7] - near_price
                newcoin = result[19] + getnearspninfo.get()
                sql = '''
                    UPDATE user_data
                    SET money=?, near=?
                    WHERE id=?
                '''
                cursor.execute(sql, [money, newcoin, select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Purchasing is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE username=? and
password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                    result = cursor.fetchone()
                    getnft_fn()
                    getnear_spn.delete(0, END)
                    getnear_spn.insert(0,0)
                    getnear_ent.delete(0, END)
                if status_near == 'no' :
                    messagebox.showinfo('Cryptonite : Get Near', 'The
Payment was cancelled.')
                else :

```

```

    messagebox.showwarning('Cryptonite : Warnning', 'You do not
have enough credit\nyou need to add credit')
    status_addcredit = messagebox.askquestion('Cryptonite : Add
Credit', 'Do you want to Add Credit?')
    if status_addcredit == 'yes' :
        addcredit_fn()
    else:
        getnft_fn()
        getnear_spn.delete(0, END)
        getnear_spn.insert(0,0)
        getnear_ent.delete(0, END)

def getNFT_monalisa():#ฝั่งขวาของฝั่งซ้าย sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 200000
    nftthb_price = nftnear_price * near
    nft_name = 'Monalisa'
    nft_status = 1

    Label(getnft_fright,image=img_mona,bg='#E8F0F2').grid(row=0,columns
pan=2,sticky='news')
    Label(getnft_fright,text='Monalisa\nLeonardo davince',bg='#E8F0F2',
font='Vardana 12').grid(row=1,columnspan=2,sticky='news')

    Label(getnft_fright,text='Near',bg='#E8F0F2').grid(row=2,column=0,s
ticky='e')
    near_ent = Entry(getnft_fright,width=8, justify=RIGHT)
    near_ent.grid(row=2,column=0,sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

    Label(getnft_fright,text='THB',bg='#E8F0F2').grid(row=3,column=0,st
icky='e')
    thb_ent = Entry(getnft_fright,width=8, justify=RIGHT)
    thb_ent.grid(row=3,column=0,sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(getnft_fright, image=btn_getnft, bd=0,
command=getbuynft).grid(row=4,column=0,sticky='w', padx=5)

def getNFT_3dboy():#ฝั่งขวาของฝั่งซ้าย sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 30701
    nftthb_price = nftnear_price * near
    nft_name = '3DBoy'
    nft_status = 2

```

```

    Label(getnft_fright,image=img_nft3D, bg='#E8F0F2').grid(row=0,column
span=2,sticky='news')
    Label(getnft_fright,text='3DBoy\nMagnum Farose',bg='#E8F0F2',
font='Vardana 12').grid(row=1,columnspan=2,sticky='news')

    Label(getnft_fright,text='Near',bg='#E8F0F2').grid(row=2,column=0,s
ticker='e')
    near_ent = Entry(getnft_fright,width=8, justify=RIGHT)
    near_ent.grid(row=2,column=0,sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

    Label(getnft_fright,text='THB',bg='#E8F0F2').grid(row=3,column=0,st
icky='e')
    thb_ent = Entry(getnft_fright,width=8, justify=RIGHT)
    thb_ent.grid(row=3,column=0,sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(getnft_fright, image=btn_getnft, bd=0,
command=getbuynft).grid(row=4,column=0,sticky='w', padx=5)

def getNFT_heath():#ฟังก์ชัน sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 24434
    nftthb_price = nftnear_price * near
    nft_name = 'Heathead'
    nft_status = 3

    Label(getnft_fright,image=img_heath, bg='#E8F0F2').grid(row=0,column
span=2,sticky='news')
    Label(getnft_fright,text='Heathead\nnrisz_kp',bg='#E8F0F2',
font='Vardana 12').grid(row=1,columnspan=2,sticky='news')

    Label(getnft_fright,text='Near',bg='#E8F0F2').grid(row=2,column=0,s
ticker='e')
    near_ent = Entry(getnft_fright,width=8, justify=RIGHT)
    near_ent.grid(row=2,column=0,sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

    Label(getnft_fright,text='THB',bg='#E8F0F2').grid(row=3,column=0,st
icky='e')
    thb_ent = Entry(getnft_fright,width=8, justify=RIGHT)
    thb_ent.grid(row=3,column=0,sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(getnft_fright, image=btn_getnft, bd=0,
command=getbuynft).grid(row=4,column=0,sticky='w', padx=5)

```

```

def getNFT_anti():#ฟังก์ชัน sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 3200
    nftthb_price = nftnear_price * near
    nft_name = 'Anticoding'
    nft_status = 4

    Label(getnft_fright,image=img_anti,bg="#E8F0F2").grid(row=0,columnspan=2,sticky='news')
    Label(getnft_fright,text='Anticoding\nDemonio',bg="#E8F0F2",font='Vardana 12').grid(row=1,columnspan=2,sticky='news')

    Label(getnft_fright,text='Near',bg="#E8F0F2").grid(row=2,column=0,sticky='e')
    near_ent = Entry(getnft_fright,width=8, justify=RIGHT)
    near_ent.grid(row=2,column=0,sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

    Label(getnft_fright,text='THB',bg="#E8F0F2").grid(row=3,column=0,sticky='e')
    thb_ent = Entry(getnft_fright,width=8, justify=RIGHT)
    thb_ent.grid(row=3,column=0,sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(getnft_fright, image=btn_getnft, bd=0,
command=getbuynft).grid(row=4,column=0,sticky='w', padx=5)

def getNFT_lotus():#ฟังก์ชัน sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 6175
    nftthb_price = nftnear_price * near
    nft_name = 'Lotus'
    nft_status = 5

    Label(getnft_fright,image=img_lotus,bg="#E8F0F2").grid(row=0,columnspan=2,sticky='news')
    Label(getnft_fright,text='Lotus\nTesco',bg="#E8F0F2", font='Vardana 12').grid(row=1,columnspan=2,sticky='news')

    Label(getnft_fright,text='Near',bg="#E8F0F2").grid(row=2,column=0,sticky='e')
    near_ent = Entry(getnft_fright,width=8, justify=RIGHT)
    near_ent.grid(row=2,column=0,sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

```

```

    Label(getnft_fright, text='THB', bg='#E8F0F2').grid(row=3, column=0, sticky='e')
    thb_ent = Entry(getnft_fright, width=8, justify=RIGHT)
    thb_ent.grid(row=3, column=0, sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(getnft_fright, image=btn_getnft, bd=0,
           command=getbuynft).grid(row=4, column=0, sticky='w', padx=5)

def getNFT_boredm():#ផ្លូវលក់ដំឡើង sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 123456
    nftthb_price = nftnear_price * near
    nft_name = 'BoredM'
    nft_status = 6

    Label(getnft_fright, image=img_bored, bg='#E8F0F2').grid(row=0, columnspan=2, sticky='news')
    Label(getnft_fright, text='BoredM\nPeter Parker', bg='#E8F0F2',
          font='Vardana 12').grid(row=1, columnspan=2, sticky='news')

    Label(getnft_fright, text='Near', bg='#E8F0F2').grid(row=2, column=0, sticky='e')
    near_ent = Entry(getnft_fright, width=8, justify=RIGHT)
    near_ent.grid(row=2, column=0, sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

    Label(getnft_fright, text='THB', bg='#E8F0F2').grid(row=3, column=0, sticky='e')
    thb_ent = Entry(getnft_fright, width=8, justify=RIGHT)
    thb_ent.grid(row=3, column=0, sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(getnft_fright, image=btn_getnft, bd=0,
           command=getbuynft).grid(row=4, column=0, sticky='w', padx=5)

def getbuynft():
    global result
    select_user = result[0]
    if nft_status == 1 :
        if result[13] == 0 :
            nft_code = 1
            status = messagebox.askquestion('Cryptonite : Get NFT',
                'Are you sure to Get %s %i %s'%(nft_name, nftnear_price, 'Near'))
            if status == 'yes' :
                payment = nftnear_price

```

```

        if result[19] >= payment :
            status_payment = messagebox.askquestion('Cryptonite : Confirm Purchasing', 'Confirm to paid %i %s'%(payment, 'Near'))
            if status_payment == 'yes' :
                near = result[19] - payment
                send_nft = result[13] + nft_code
                sql = '''
                    UPDATE user_data
                    SET nft1=? , near=?
                    WHERE id=?
                    ...
                '''
                cursor.execute(sql, [send_nft, near,
select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite : Successfully', 'Your Purchasing is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE username=? and password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                    result = cursor.fetchone()
                    getnft_fn()
                    if status_payment == 'no' :
                        messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Warnning', 'You do not have enough Near\nyou need to Get Near')
                        status_getnear = messagebox.askquestion('Cryptonite : Get Near', 'Do you want to Get Near?')
                        if status_getnear == 'yes' :
                            getnft_fn()
                        else:
                            messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Get NFT', 'This NFT is Sold Out.')
                elif nft_status == 2 :
                    if result[14] == 0 :
                        nft_code = 1
                        status = messagebox.askquestion('Cryptonite : Get NFT', 'Are you sure to Get %s %i %s'%(nft_name, nftnear_price,'Near'))
                        if status == 'yes' :
                            payment = nftnear_price

```

```

        if result[19] >= payment :
            status_payment = messagebox.askquestion('Cryptonite : Confirm Purchasing', 'Confirm to paid %i %s'%(payment, 'Near'))
            if status_payment == 'yes' :
                near = result[19] - payment
                send_nft = result[14] + nft_code
                sql = '''
                    UPDATE user_data
                    SET nft2=? , near=?
                    WHERE id=?
                    ...
                '''
                cursor.execute(sql, [send_nft, near,
select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite : Successfully', 'Your Purchasing is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE username=? and password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                    result = cursor.fetchone()
                    getnft_fn()
                    if status_payment == 'no' :
                        messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Warnning', 'You do not have enough Near\nyou need to Get Near')
                        status_getnear = messagebox.askquestion('Cryptonite : Get Near', 'Do you want to Get Near?')
                        if status_getnear == 'yes' :
                            getnft_fn()
                        else:
                            messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Get NFT', 'This NFT is Sold Out.')
                elif nft_status == 3 :
                    if result[15] == 0 :
                        nft_code = 1
                        status = messagebox.askquestion('Cryptonite : Get NFT', 'Are you sure to Get %s %i %s'%(nft_name, nftnear_price,'Near'))
                        if status == 'yes' :
                            payment = nftnear_price

```

```

        if result[19] >= payment :
            status_payment = messagebox.askquestion('Cryptonite : Confirm Purchasing', 'Confirm to paid %i %s'%(payment, 'Near'))
            if status_payment == 'yes' :
                near = result[19] - payment
                send_nft = result[15] + nft_code
                sql = '''
                    UPDATE user_data
                    SET nft3=? , near=?
                    WHERE id=?
                    ...
                '''
                cursor.execute(sql, [send_nft, near,
select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite : Successfully', 'Your Purchasing is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE username=? and password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                    result = cursor.fetchone()
                    getnft_fn()
                    if status_payment == 'no' :
                        messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Warnning', 'You do not have enough Near\nyou need to Get Near')
                        status_getnear = messagebox.askquestion('Cryptonite : Get Near', 'Do you want to Get Near?')
                        if status_getnear == 'yes' :
                            getnft_fn()
                        else:
                            messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Get NFT', 'This NFT is Sold Out.')
                elif nft_status == 4 :
                    if result[16] == 0 :
                        nft_code = 1
                        status = messagebox.askquestion('Cryptonite : Get NFT', 'Are you sure to Get %s %i %s'%(nft_name, nftnear_price,'Near'))
                        if status == 'yes' :
                            payment = nftnear_price

```

```

        if result[19] >= payment :
            status_payment = messagebox.askquestion('Cryptonite : Confirm Purchasing', 'Confirm to paid %i %s'%(payment, 'Near'))
            if status_payment == 'yes' :
                near = result[19] - payment
                send_nft = result[16] + nft_code
                sql = '''
                    UPDATE user_data
                    SET nft4=?, near=?
                    WHERE id=?
                    ...
                '''
                cursor.execute(sql, [send_nft, near,
select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite : Successfully', 'Your Purchasing is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE username=? and password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                    result = cursor.fetchone()
                    getnft_fn()
                    if status_payment == 'no' :
                        messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Warnning', 'You do not have enough Near\nyou need to Get Near')
                        status_getnear = messagebox.askquestion('Cryptonite : Get Near', 'Do you want to Get Near?')
                        if status_getnear == 'yes' :
                            getnft_fn()
                        else:
                            messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Get NFT', 'This NFT is Sold Out.')
                elif nft_status == 5 :
                    if result[17] == 0 :
                        nft_code = 1
                        status = messagebox.askquestion('Cryptonite : Get NFT', 'Are you sure to Get %s %i %s'%(nft_name, nftnear_price,'Near'))
                        if status == 'yes' :
                            payment = nftnear_price

```

```

        if result[19] >= payment :
            status_payment = messagebox.askquestion('Cryptonite : Confirm Purchasing', 'Confirm to paid %i %s'%(payment, 'Near'))
            if status_payment == 'yes' :
                near = result[19] - payment
                send_nft = result[17] + nft_code
                sql = '''
                    UPDATE user_data
                    SET nft5=?, near=?
                    WHERE id=?
                    ...
                '''
                cursor.execute(sql, [send_nft, near,
select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite : Successfully', 'Your Purchasing is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE
username=? and password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.
get()])
                    result = cursor.fetchone()
                    getnft_fn()
                    if status_payment == 'no' :
                        messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Warnning', 'You do not have enough Near\nyou need to Get Near')
                        status_getnear = messagebox.askquestion('Cryptonite : Get Near', 'Do you want to Get Near?')
                        if status_getnear == 'yes' :
                            getnft_fn()
                        else:
                            messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Get NFT', 'This NFT is Sold Out.')
                elif nft_status == 6 :
                    if result[18] == 0 :
                        nft_code = 1
                        status = messagebox.askquestion('Cryptonite : Get NFT ','Are you sure to Get %s %i %s'%(nft_name, nftnear_price,'Near'))
                        if status == 'yes' :
                            payment = nftnear_price

```

```

        if result[19] >= payment :
            status_payment = messagebox.askquestion('Cryptonite : Confirm Purchasing', 'Confirm to paid %i %s%(payment, 'Near')
            if status_payment == 'yes' :
                near = result[19] - payment
                send_nft = result[18] + nft_code
                sql = '''
                    UPDATE user_data
                    SET nft6=? , near=?
                    WHERE id=?
                    ...
                    cursor.execute(sql, [send_nft, near,
select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite : Successfully', 'Your Purchasing is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE
username=? and password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.
get()])
                    result = cursor.fetchone()
                    getnft_fn()
                    if status_payment == 'no' :
                        messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Warnning', 'You do not have enough Near\nyou need to Get Near')
                        status_getnear = messagebox.askquestion('Cryptonite : Get Near', 'Do you want to Get Near?')
                        if status_getnear == 'yes' :
                            getnft_fn()
                        else:
                            messagebox.showinfo('Cryptonite : Get NFT', 'The Payment was cancelled.')
                    else :
                        messagebox.showwarning('Cryptonite : Get NFT', 'This NFT is Sold Out.')

def sellcoin_fn():
    global status_login, total_coin, sellcoinoption, sellcoin_spn,
sellcoin_ent, mycointosell
    if status_login == False :
        messagebox.showwarning('Cryptonite : You must login first.', 'You are not logged in, You must login first')

```

```

        backtologin()
else:
    status_login = True
    root.title('Cryptonite : Sell Coin %s%%(%'*150, fullname))
    global sellcoin_ftop,sellcoin_fcenter,sellcoin_fbottom,
sellcoin_f
    global sellcoin_ent,sellcoin_spn,sellcoin_ent
    global findoption

    sellcoin_f = Frame(root)
    sellcoin_f.rowconfigure((0,1,2,3), weight=1)
    sellcoin_f.columnconfigure((0,1,2,3), weight=1)
    sellcoin_f.place(x=0, y=0, width=w, height=h)

    sellcoin_ftop = Frame(sellcoin_f, bg="white")
    sellcoin_ftop.rowconfigure((0,1,2,3),weight=1)
    sellcoin_ftop.columnconfigure((0,1,2,3,4),weight=1)
    sellcoin_ftop.grid(row=0,column=0,columnspan=4,sticky='news')

    sellcoin_fcenter = Frame(sellcoin_f, bg="#D1D1D1")
    sellcoin_fcenter.rowconfigure((0,1,2,3,4,5,6,7,8,9),weight=1)
    sellcoin_fcenter.columnconfigure((0,1,2,3,4,5,6,7,8,9),weight=1)
)
    sellcoin_fcenter.grid(row=1,column=0,rowspan=6,columnspan=5,sticky='news')

    sellcoin_fright = Frame(sellcoin_f,bg='pink')
    sellcoin_fright.rowconfigure((0,1,2,3,4),weight=1)
    sellcoin_fright.columnconfigure((0,1),weight=1)
    sellcoin_fright.grid(row=1,column=2,sticky='news',rowspan=4,columnspan=2)

    sellcoin_fbottom = Frame(sellcoin_f, bg="pink")
    bg = Label(sellcoin_fbottom,image=login_bg)
    bg.place(x=0,y=0,width=w,height=h)
    sellcoin_fbottom.rowconfigure((0,1),weight=1)
    sellcoin_fbottom.columnconfigure((0,1,2,3,4),weight=1)
    sellcoin_fbottom.grid(row=5,column=0,columnspan=4,sticky='news')
)

#ส่วนบน
    Label(sellcoin_ftop,text='Sell Coin',bg='white',font='Vanada 20 bold').grid(row=1,column=0,sticky='w',padx=10)
    Label(sellcoin_ftop,text='Total',bg='white',font='Vanada 20 bold').grid(row=1,column=1,sticky='w')

```

```

        Label.sellcoin_ftop.text=( '%0.2f'%(result[7])),bg='white',font=
'Verdana 15 bold').grid(row=1,column=2,sticky='w')

        Label.sellcoin_ftop.text='THB',bg='white',font='Vanada 20
bold').grid(row=1,column=2,columnspan=2)
        Button.sellcoin_ftop, image=btn_marketcap, bg='white',
bd=0,command=margetcap_fn).grid(row=1,column=3,columnspan=2,ipadx=10,ip
ady=10)

#ส่วนคลาย
findoption = StringVar()
findoption.set("Select Coin")
sellcoinoption = OptionMenu.sellcoin_fcenter,findoption,
*mycoin)
sellcoinoption.grid(row=1,column=0,sticky=E)

Label.sellcoin_fcenter, text='Your Coin',font='Vanada 20 bold',
bg='#D1D1D1').grid(row=0, column=1, sticky=W)
mycointosell = Entry.sellcoin_fcenter, width=9, justify=RIGHT)
mycointosell.grid(row=0, column=1, sticky=E)
total_coin = Entry.sellcoin_fcenter, justify=RIGHT, width=15)
total_coin.grid(row=1, column=1, sticky=E)
Button.sellcoin_fcenter, image=ico_search, bd=0, bg='#D1D1D1',
command=searchandsellcrypto).grid(row=1, column=2, sticky=W)
Label.sellcoin_fcenter, text='Amout', bg='#D1D1D1',
font='Verdana 12').grid(row=2, column=2, sticky=W, padx=5)
sellcoin_spn = Spinbox.sellcoin_fcenter,from_=0,to=10,width=7,
justify=RIGHT, textvariable=spincoinsell_info)
sellcoin_spn.grid(row=2,column=1,sticky='e')
Label.sellcoin_fcenter, text='THB', bg='#D1D1D1', font='Verdana
12').grid(row=3, column=2, sticky=W, padx=5)
sellcoin_ent = Entry.sellcoin_fcenter, width=15,
justify=RIGHT) #ร่องเงิน2
sellcoin_ent.grid(row=3,column=1,sticky='ne')
Button.sellcoin_fcenter,image=btn_sellcoin, bd=0, bg='#D1D1D1',
command=sellcoin).grid(row=4,column=1,ipadx=7,ipady=7,sticky='ne')
Button.sellcoin_fcenter, image=btn_getcoin, bg='#D1D1D1',
bd=0,command=getcoin_fn).grid(row=5,column=1,ipadx=7,ipady=7,sticky='ne
')

#จุดทางขวา
Label.sellcoin_fright,image=img_bitcoin,bg='#D1D1D1').grid(row=
0,column=0,columnspan=2,sticky='news')
Label.sellcoin_fright,image=img_doge,bg='#D1D1D1').grid(row=1,c
olumn=0,columnspan=2,sticky='news')

```

```

    Label.sellcoin_fright,image=img_eth, bg='#D1D1D1').grid(row=2,ro
wspan=3,column=0,columnspan=2,sticky='news')

    #ส่วนล่าง
    Button.sellcoin_fbottom ,image=ico_return, compound=LEFT,
text='Back to
Profile',command=profile_fn).grid(row=0,column=0,ipady=10,ipadx=10,sti
cky='w')
    Button.sellcoin_fbottom,image=ico_reload,text='Refresh',compoun
d=RIGHT, command=sellcoin_fn).grid(row=0,column=5,ipady=10,ipadx=10)

def searchandsellcrypto() :
    total_coin.delete(0, END)
    sellcoin_ent.delete(0, END)
    mycointosell.delete(0, END)
    get = 0
    if findoption.get() == 'BTC' :
        get = btc_coin * spincoinsell_info.get()
        total_coin.insert(0, btc_coin)
        sellcoin_ent.insert(0, get)
        mycointosell.insert(0, result[8])
    elif findoption.get() == 'ETH' :
        get = eth_coin * spincoinsell_info.get()
        total_coin.insert(0, eth_coin)
        sellcoin_ent.insert(0, get)
        mycointosell.insert(0, result[9])
    elif findoption.get() == 'DOGE' :
        get = doge_coin * spincoinsell_info.get()
        total_coin.insert(0, doge_coin)
        sellcoin_ent.insert(0, get)
        mycointosell.insert(0, result[10])
    elif findoption.get() == 'KUB' :
        get = kub_coin * spincoinsell_info.get()
        total_coin.insert(0, kub_coin)
        sellcoin_ent.insert(0, get)
        mycointosell.insert(0, result[11])
    elif findoption.get() == 'BNB' :
        get = bnb_coin * spincoinsell_info.get()
        total_coin.insert(0, bnb_coin)
        sellcoin_ent.insert(0, get)
        mycointosell.insert(0, result[12])
def sellcoin():
    global result

```

```

    status = messagebox.askquestion('Cryptonite : Sell Coin ','Are you
sure to Sell your Coin %d %s'%(spincoinsell_info.get(),
findoption.get()))
    if status == 'yes' :
        print(findoption.get())
        if findoption.get() == 'BTC' :
            get = btc_coin * spincoinsell_info.get()
            if result[8] >= spincoinsell_info.get() :
                status_get = messagebox.askquestion('Cryptonite :
Confirm Sell Coin', 'Confirm to get %i'%(get))
                if status_get == 'yes':
                    select_user = result[0]
                    money = result[7] + get
                    newcoin = result[8] - spincoinsell_info.get()
                    sql = '''
                        UPDATE user_data
                        SET money=?, btc=?
                        WHERE id=?
                    '''
                    ...
                    cursor.execute(sql, [money, newcoin, select_user])
                    conn.commit()
                    update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Sales is Successfully.')
                    if update:
                        sql = "SELECT * FROM user_data WHERE username=?"
                        and password=?"
                        cursor.execute(sql,[userinfo.get(),pwdinfo.get(
)])
                        result = cursor.fetchone()
                        sellcoin_fn()
                        sellcoin_spn.delete(0, END)
                        sellcoin_spn.insert(0,0)
                        mycointosell.delete(0, END)
                        total_coin.delete(0, END)
                        findoption.set("Select Coin")
                    if status_get == 'no' :
                        messagebox.showinfo('Cryptonite : Sell Coin', 'Sell
Coin was cancelled.')
                        sellcoin_spn.delete(0, END)
                        sellcoin_spn.insert(0,0)
                        mycointosell.delete(0, END)
                        total_coin.delete(0, END)
                        findoption.set("Select Coin")
                else :
                    messagebox.showwarning('Cryptonite : Sell Coin', 'You
do not have enough Coin\nyou need to Get Coin')

```

```

        status_getcoin = messagebox.askquestion('Cryptonite :
Add Credit', 'Do you want to Add Credit?')
        if status_getcoin == 'yes' :
            getcoin_fn()
        else :
            sellcoin_fn()
            sellcoin_spn.delete(0, END)
            sellcoin_spn.insert(0,0)
            mycointosell.delete(0, END)
            total_coin.delete(0, END)
            findoption.set("Select Coin")
            messagebox.showinfo('Cryptonite : Sell Coin', 'Sell
Coin was cancelled.')
        if findoption.get() == 'ETH' :
            get = eth_coin * spincoinsell_info.get()
            if result[9] >= spincoinsell_info.get() :
                status_get = messagebox.askquestion('Cryptonite :
Confirm Sell Coin', 'Confirm to get %i'%(get))
                if status_get == 'yes':
                    select_user = result[0]
                    money = result[7] + get
                    newcoin = result[9] - spincoinsell_info.get()
                    sql = '''
                        UPDATE user_data
                        SET money=?, eth=?
                        WHERE id=?
                    '''
                    cursor.execute(sql, [money, newcoin, select_user])
                    conn.commit()
                    update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Sales is Successfully.')
                    if update:
                        sql = "SELECT * FROM user_data WHERE username=?"
                        and password=?"
                        cursor.execute(sql,[userinfo.get(),pwdinfo.get(
)])
                        result = cursor.fetchone()
                        sellcoin_fn()
                        sellcoin_spn.delete(0, END)
                        sellcoin_spn.insert(0,0)
                        mycointosell.delete(0, END)
                        total_coin.delete(0, END)
                        findoption.set("Select Coin")
                    if status_get == 'no' :
                        messagebox.showinfo('Cryptonite : Sell Coin', 'Sell
Coin was cancelled.')

```

```

        sellcoin_spn.delete(0, END)
        sellcoin_spn.insert(0,0)
        mycointosell.delete(0, END)
        total_coin.delete(0, END)
        findoption.set("Select Coin")

    else :
        messagebox.showwarning('Cryptonite : Sell Coin', 'You
do not have enough Coin\nyou need to Get Coin')
        status_getcoin = messagebox.askquestion('Cryptonite :
Add Credit', 'Do you want to Add Credit?')
        if status_getcoin == 'yes' :
            getcoin_fn()
        else :
            sellcoin_fn()
            sellcoin_spn.delete(0, END)
            sellcoin_spn.insert(0,0)
            mycointosell.delete(0, END)
            total_coin.delete(0, END)
            findoption.set("Select Coin")

    if findoption.get() == 'DOGE' :
        get = doge_coin * spincoinsell_info.get()
        if result[10] >= spincoinsell_info.get() :
            status_get = messagebox.askquestion('Cryptonite :
Confirm Sell Coin', 'Confirm to get %i'%(get))
            if status_get == 'yes':
                select_user = result[0]
                money = result[7] + get
                newcoin = result[10] - spincoinsell_info.get()
                sql = '''
                    UPDATE user_data
                    SET money=?, doge=?
                    WHERE id=?
                    ...
                '''
                cursor.execute(sql, [money, newcoin, select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Sales is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE username=?"
                    and password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.get(
                    )])
                    result = cursor.fetchone()
                    sellcoin_fn()
                    sellcoin_spn.delete(0, END)
                    sellcoin_spn.insert(0,0)

```

```

        mycointosell.delete(0, END)
        total_coin.delete(0, END)
        findoption.set("Select Coin")
    if status_get == 'no' :
        messagebox.showinfo('Cryptonite : Sell Coin', 'Sell
Coin was cancelled.')
        sellcoin_spn.delete(0, END)
        sellcoin_spn.insert(0,0)
        mycointosell.delete(0, END)
        total_coin.delete(0, END)
        findoption.set("Select Coin")
    else :
        messagebox.showwarning('Cryptonite : Sell Coin', 'You
do not have enough Coin\nyou need to Get Coin')
        status_getcoin = messagebox.askquestion('Cryptonite :
Add Credit', 'Do you want to Add Credit?')
        if status_getcoin == 'yes' :
            getcoin_fn()
        else :
            sellcoin_fn()
            sellcoin_spn.delete(0, END)
            sellcoin_spn.insert(0,0)
            mycointosell.delete(0, END)
            total_coin.delete(0, END)
            findoption.set("Select Coin")
    if findoption.get() == 'KUB' :
        get = kub_coin * spincoinsell_info.get()
        if result[11] >= spincoinsell_info.get() :
            status_get = messagebox.askquestion('Cryptonite :
Confirm Sell Coin', 'Confirm to get %i'%(get))
            if status_get == 'yes':
                select_user = result[0]
                money = result[7] + get
                newcoin = result[11] - spincoinsell_info.get()
                sql = '''
                    UPDATE user_data
                    SET money=?, kub=?
                    WHERE id=?
                '''
                cursor.execute(sql, [money, newcoin, select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Sales is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE username=?"
                    and password=?"

```

```

        cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
    )
    result = cursor.fetchone()
    sellcoin_fn()
    sellcoin_spn.delete(0, END)
    sellcoin_spn.insert(0,0)
    mycointosell.delete(0, END)
    total_coin.delete(0, END)
    findoption.set("Select Coin")
    if status_get == 'no' :
        messagebox.showinfo('Cryptonite : Sell Coin', 'Sell
Coin was cancelled.')
        sellcoin_spn.delete(0, END)
        sellcoin_spn.insert(0,0)
        mycointosell.delete(0, END)
        total_coin.delete(0, END)
        findoption.set("Select Coin")
    else :
        messagebox.showwarning('Cryptonite : Sell Coin', 'You
do not have enough Coin\nyou need to Get Coin')
        status_getcoin = messagebox.askquestion('Cryptonite :
Add Credit', 'Do you want to Add Credit?')
        if status_getcoin == 'yes' :
            getcoin_fn()
        else :
            sellcoin_fn()
            sellcoin_spn.delete(0, END)
            sellcoin_spn.insert(0,0)
            mycointosell.delete(0, END)
            total_coin.delete(0, END)
            findoption.set("Select Coin")
    if findoption.get() == 'BNB' :
        get = bnb_coin * spincoinsell_info.get()
        if result[12] >= spincoinsell_info.get() :
            status_get = messagebox.askquestion('Cryptonite :
Confirm Sell Coin', 'Confirm to get %i'%(get))
            if status_get == 'yes':
                select_user = result[0]
                money = result[7] + get
                newcoin = result[12] - spincoinsell_info.get()
                sql = '''
                    UPDATE user_data
                    SET money=?, bnb=?
                    WHERE id=?
                '''
                cursor.execute(sql, [money, newcoin, select_user])

```

```

        conn.commit()
        update = messagebox.showinfo('Cryptonite : Successfully', 'Your Sales is Successfully.')
        if update:
            sql = "SELECT * FROM user_data WHERE username=? and password=?"
            cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
            result = cursor.fetchone()
            sellcoin_fn()
            sellcoin_spn.delete(0, END)
            sellcoin_spn.insert(0,0)
            mycointosell.delete(0, END)
            total_coin.delete(0, END)
            findoption.set("Select Coin")
        if status_get == 'no' :
            messagebox.showinfo('Cryptonite : Sell Coin', 'Sell Coin was cancelled.')
            sellcoin_spn.delete(0, END)
            sellcoin_spn.insert(0,0)
            mycointosell.delete(0, END)
            total_coin.delete(0, END)
            findoption.set("Select Coin")
        else :
            messagebox.showwarning('Cryptonite : Sell Coin', 'You do not have enough Coin\nyou need to Get Coin')
            status_getcoin = messagebox.askquestion('Cryptonite : Add Credit', 'Do you want to Add Credit?')
            if status_getcoin == 'yes' :
                getcoin_fn()
            else :
                sellcoin_fn()
                sellcoin_spn.delete(0, END)
                sellcoin_spn.insert(0,0)
                mycointosell.delete(0, END)
                total_coin.delete(0, END)
                findoption.set("Select Coin")
def sellnft_fn():
    global status_login, near
    near = 352.09
    if status_login == False :
        messagebox.showwarning('Cryptonite : You must login first.', 'You are not logged in, You must login first')
        backtologin()
    else:
        status_login = True

```

```

global snft_f
root.title('Cryptonite : Sell NFT %s%s'%' '*151, fullname)
snft_f = Frame(root, bg="orange")
snft_f.rowconfigure((0,1,2,3,4), weight=1)
snft_f.columnconfigure((0,1,2,3,4,5), weight=1)
snft_f.place(x=0, y=0, width=w, height=h)

global snft_ftop,snft_fleft,snft_fleft,snft_fright,snft_fbottom
snft_ftop = Frame(snft_f, bg='white')
snft_ftop.rowconfigure((0,1,2),weight=1)
snft_ftop.columnconfigure((0,1,2,3,4),weight=1)
snft_ftop.grid(row=0,column=0,columnspan=6,sticky='news')

snft_fleft = Frame(snft_f, bg='#D1D1D1')
snft_fleft.rowconfigure((0,1),weight=1)
snft_fleft.columnconfigure((0,1,2),weight=1)
snft_fleft.grid(row=1,column=0,rowspan=5,columnspan=4,sticky='news')
)

snft_fright = Frame(snft_f, bg="#E8F0F2")
snft_fright.rowconfigure((0,1,2,3,4,5),weight=1)
snft_fright.columnconfigure((0,1,2),weight=1)
snft_fright.grid(row=1,column=4,rowspan=5,columnspan=2,sticky='news')
')

snft_fbottom = Frame(snft_f, bg='white')
bg = Label(snft_fbottom,image=login_bg)
bg.place(x=0,y=0,width=w,height=h)
snft_fbottom.rowconfigure((0,1,2),weight=1)
snft_fbottom.columnconfigure((0,1,2,3,4),weight=1)
snft_fbottom.grid(row=6,column=0,columnspan=6,sticky='news')

#ล้ำบนบัน
Label(snft_ftop,text='Sell NFT',bg='white',font='Vadana 20 bold').grid(row=1,column=0,sticky='w')
Label(snft_ftop,text='Total',bg='white',font='Verdana 20 bold').grid(row=1,column=1,sticky='w')
Label(snft_ftop,text='%.2f'%(result[7])),bg='white',font='Verdana 20 bold').grid(row=1,column=2,sticky='w')
Label(snft_ftop,text='THB',bg='white',font='Verdana 20 bold').grid(row=1,column=3,sticky='w')
Button(snft_ftop,image=btn_marketcap,bg='white',bd=0,command=addredit_fn).grid(row=1,column=4)

#ล้ำบนช้าๆ

```

```

    if result[13] == 0 :
        Button(snft_fleft,image=img.blurmonalisa,compound=TOP,command=sellNFT_monalisa).grid(row=0,column=0,sticky='w',ipadx=30,ipady=30)
    else :
        Button(snft_fleft,image=img.mona,text='Monalisa',compound=TOP,command=sellNFT_monalisa).grid(row=0,column=0,sticky='w',ipadx=30,ipady=30)
    if result[14] == 0 :
        Button(snft_fleft,image=img.blur3dboy,command=sellNFT_3dboy).grid(row=0,column=1,sticky='w',ipadx=30,ipady=30)
    else :
        Button(snft_fleft,image=img.nft3D,text='3DBoy',compound=TOP,command=sellNFT_3dboy).grid(row=0,column=1,sticky='w',ipadx=30,ipady=30)
    if result[15] == 0 :
        Button(snft_fleft,image=img.blurheathead,command=sellNFT_heath).grid(row=0,column=2,sticky='w',ipadx=30,ipady=30)
    else :
        Button(snft_fleft,image=img.heath,text='Heathead',compound=TOP,command=sellNFT_heath).grid(row=0,column=2,sticky='w',ipadx=30,ipady=30)
    if result[16] == 0 :
        Button(snft_fleft,image=img.bluranti,command=sellNFT_anti).grid(row=1,column=0,sticky='w',ipadx=30,ipady=30)
    else :
        Button(snft_fleft,image=img.anti,text='Anticoding',compound=TOP,command=sellNFT_anti).grid(row=1,column=0,sticky='w',ipadx=30,ipady=30)
    if result[17] == 0 :
        Button(snft_fleft,image=img.blurlotus,command=sellNFT_lotus).grid(row=1,column=1,sticky='w',ipadx=30,ipady=30)
    else:
        Button(snft_fleft,image=img.lotus,text='Lotus',compound=TOP,command=sellNFT_lotus).grid(row=1,column=1,sticky='w',ipadx=30,ipady=30)
    if result[18] == 0 :
        Button(snft_fleft,image=img.blurboredm,command=sellNFT_boredm).grid(row=1,column=2,sticky='w',ipadx=30,ipady=30)
    else :
        Button(snft_fleft,image=img.bored,text='BoredM',compound=TOP,command=sellNFT_boredm).grid(row=1,column=2,sticky='w',ipadx=30,ipady=30)

#ล้านล่าง
Button(snft_fbottom, image=ico_return, compound=LEFT,text='Back To Profile',
command=profile_fn).grid(row=3,column=0,sticky='w',ipadx=10,ipady=10)
Button(snft_fbottom,image=ico_reload,text='Refresh',compound=LEFT,
command=sellnft_fn).grid(row=3,column=4,sticky='e',ipadx=10,ipady=10)

```

```

def sellNFT_monalisa():#ฟังก์ชัน sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 200000
    nftthb_price = nftnear_price * near
    nft_name = 'Monalisa'
    nft_status = 1

    Label(snft_fright,image=img_mona,bg="#E8F0F2").grid(row=0,columnspan=2,sticky='news')
    Label(snft_fright,text='Monalisa\nLeonardo davince',bg="#E8F0F2",font='Vardana 12').grid(row=1,columnspan=2,sticky='news')

    Label(snft_fright,text='Near',bg="#E8F0F2").grid(row=2,column=0,sticky='e')
    near_ent = Entry(snft_fright,width=8, justify=RIGHT)
    near_ent.grid(row=2,column=0,sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

    Label(snft_fright,text='THB',bg="#E8F0F2").grid(row=3,column=0,sticky='e')
    thb_ent = Entry(snft_fright,width=8, justify=RIGHT)
    thb_ent.grid(row=3,column=0,sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(snft_fright, image=btn_sellnft, bd=0,
command=sellnft).grid(row=4,column=0,sticky='w', padx=5)

def sellNFT_3dboy():#ฟังก์ชัน sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 30701
    nftthb_price = nftnear_price * near
    nft_name = '3DBoy'
    nft_status = 2

    Label(snft_fright,image=img_nft3D,bg="#E8F0F2").grid(row=0,columnspan=2,sticky='news')
    Label(snft_fright,text='3DBoy\nMagnum Farose',bg="#E8F0F2",font='Vardana 12').grid(row=1,columnspan=2,sticky='news')

    Label(snft_fright,text='Near',bg="#E8F0F2").grid(row=2,column=0,sticky='e')
    near_ent = Entry(snft_fright,width=8, justify=RIGHT)
    near_ent.grid(row=2,column=0,sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

```

```

    Label(snft_fright, text='THB', bg='#E8F0F2').grid(row=3, column=0, sticky='e')
    thb_ent = Entry(snft_fright, width=8, justify=RIGHT)
    thb_ent.grid(row=3, column=0, sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(snft_fright, image=btn_sellnft, bd=0,
           command=sellnft).grid(row=4, column=0, sticky='w', padx=5)

def sellNFT_heath():#ផ្លូវការលក់ខ្លួន sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 24434
    nftthb_price = nftnear_price * near
    nft_name = 'Heathead'
    nft_status = 3

    Label(snft_fright, image=img_heath, bg='#E8F0F2').grid(row=0, columnspan=2, sticky='news')
    Label(snft_fright, text='Heathead\nnrisz_kp', bg='#E8F0F2',
          font='Vardana 12').grid(row=1, columnspan=2, sticky='news')

    Label(snft_fright, text='Near', bg='#E8F0F2').grid(row=2, column=0, sticky='e')
    near_ent = Entry(snft_fright, width=8, justify=RIGHT)
    near_ent.grid(row=2, column=0, sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

    Label(snft_fright, text='THB', bg='#E8F0F2').grid(row=3, column=0, sticky='e')
    thb_ent = Entry(snft_fright, width=8, justify=RIGHT)
    thb_ent.grid(row=3, column=0, sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(snft_fright, image=btn_sellnft, bd=0,
           command=sellnft).grid(row=4, column=0, sticky='w', padx=5)

def sellNFT_anti():#ផ្លូវការលក់ខ្លួន sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 3200
    nftthb_price = nftnear_price * near
    nft_name = 'Anticoding'
    nft_status = 4

    Label(snft_fright, image=img_anti, bg='#E8F0F2').grid(row=0, columnspan=2, sticky='news')

```

```

    Label(snft_fright, text='Anticoding\nDemonio', bg='#E8F0F2',
font='Vardana 12').grid(row=1, columnspan=2, sticky='news')

    Label(snft_fright, text='Near', bg='#E8F0F2').grid(row=2, column=0, sticky='e')
    near_ent = Entry(snft_fright, width=8, justify=RIGHT)
    near_ent.grid(row=2, column=0, sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

    Label(snft_fright, text='THB', bg='#E8F0F2').grid(row=3, column=0, sticky='e')
    thb_ent = Entry(snft_fright, width=8, justify=RIGHT)
    thb_ent.grid(row=3, column=0, sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(snft_fright, image=btn_sellnft, bd=0,
command=sellnft).grid(row=4, column=0, sticky='w', padx=5)

def sellNFT_lotus():#ផ្លូវការលក់ដំឡើង sellnft_fn()
    global nft_status, nftnear_price, nftthb_price, nft_name
    nftnear_price = 6175
    nftthb_price = nftnear_price * near
    nft_name = 'Lotus'
    nft_status = 5

    Label(snft_fright, image=img_lotus, bg='#E8F0F2').grid(row=0, columnspan=2, sticky='news')
    Label(snft_fright, text='Lotus\nTesco', bg='#E8F0F2', font='Vardana 12').grid(row=1, columnspan=2, sticky='news')

    Label(snft_fright, text='Near', bg='#E8F0F2').grid(row=2, column=0, sticky='e')
    near_ent = Entry(snft_fright, width=8, justify=RIGHT)
    near_ent.grid(row=2, column=0, sticky='w', padx=5)
    near_ent.insert(0, nftnear_price)

    Label(snft_fright, text='THB', bg='#E8F0F2').grid(row=3, column=0, sticky='e')
    thb_ent = Entry(snft_fright, width=8, justify=RIGHT)
    thb_ent.grid(row=3, column=0, sticky='w', padx=5)
    thb_ent.insert(0, nftthb_price)

    Button(snft_fright, image=btn_sellnft, bd=0,
command=sellnft).grid(row=4, column=0, sticky='w', padx=5)

def sellNFT_boredm():#ផ្លូវការលក់ដំឡើង sellnft_fn()

```

```

global nft_status, nftnear_price, nftthb_price, nft_name
nftnear_price = 123456
nftthb_price = nftnear_price * near
nft_name = 'BoredM'
nft_status = 6

Label(snft_fright,image=img_bored,bg='#E8F0F2').grid(row=0,columnspan=2,sticky='news')
Label(snft_fright,text='BoredM\nPeter Parker',bg='#E8F0F2',font='Vardana 12').grid(row=1,columnspan=2,sticky='news')

Label(snft_fright,text='Near',bg='#E8F0F2').grid(row=2,column=0,sticky='e')
near_ent = Entry(snft_fright,width=8, justify=RIGHT)
near_ent.grid(row=2,column=0,sticky='w', padx=5)
near_ent.insert(0, nftnear_price)

Label(snft_fright,text='THB',bg='#E8F0F2').grid(row=3,column=0,sticky='e')
thb_ent = Entry(snft_fright,width=8, justify=RIGHT)
thb_ent.grid(row=3,column=0,sticky='w', padx=5)
thb_ent.insert(0, nftthb_price)

Button(snft_fright, image=btn_sellnft, bd=0, command=sellnft).grid(row=4,column=0,sticky='w', padx=5)
def sellnft():
    global result
    select_user = result[0]
    if nft_status == 1 :
        if result[13] == 1 :
            nft_code = 1
            status = messagebox.askquestion('Cryptonite : Sell NFT','Are you sure to Sell %s %i %s'%(nft_name, nftnear_price,'Near'))
            if status == 'yes' :
                getthb = nftthb_price
                status_payment = messagebox.askquestion('Cryptonite : Confirm Selling', 'Confirm to Get %i %s'%(getthb, 'THB'))
                if status_payment == 'yes' :
                    money = result[7] + nftthb_price
                    send_nft = result[13] - nft_code
                    sql = '''
                        UPDATE user_data
                        SET money=?, nft1=?
                        WHERE id=?
                    '''
                    cursor.execute(sql, [money, send_nft, select_user])

```

```

        conn.commit()
        update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Purchasing is Successfully.')
        if update:
            sql = "SELECT * FROM user_data WHERE username=?
and password=?"
            cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
            result = cursor.fetchone()
            sellnft_fn()
            if status_payment == 'no' :
                messagebox.showinfo('Cryptonite : Sell NFT', 'The
Selling was cancelled.')
            else :
                messagebox.showwarning('Cryptonite : Warnning', 'You do not
have this NFT\nyou need to Get NFT')
                status_getnft = messagebox.askquestion('Cryptonite : Get
NFT', 'Do you want to Get NFT?')
                if status_getnft == 'yes' :
                    getnft_fn()
                else :
                    messagebox.showinfo('Cryptonite : Sell NFT', 'The
Selling was cancelled.')
            elif nft_status == 2 :
                if result[14] == 1 :
                    nft_code = 1
                    status = messagebox.askquestion('Cryptonite : Sell NFT
','Are you sure to Sell %s %i %s'%(nft_name, nftnear_price,'Near'))
                    if status == 'yes' :
                        getthb = nftthb_price
                        status_payment = messagebox.askquestion('Cryptonite :
Confirm Selling', 'Confirm to Get %i %s'%(getthb, 'THB'))
                        if status_payment == 'yes' :
                            money = result[7] + nftthb_price
                            send_nft = result[14] - nft_code
                            sql = '''
                                UPDATE user_data
                                SET money=?, nft2=?
                                WHERE id=?
                            '''
                            cursor.execute(sql, [money, send_nft, select_user])
                            conn.commit()
                            update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Purchasing is Successfully.')
                            if update:

```



```

            sellnft_fn()
        if status_payment == 'no' :
            messagebox.showinfo('Cryptonite : Sell NFT', 'The
Selling was cancelled.')
        else :
            messagebox.showwarning('Cryptonite : Warnning', 'You do not
have this NFT\nyou need to Get NFT')
            status_getnft = messagebox.askquestion('Cryptonite : Get
NFT', 'Do you want to Get NFT?')
            if status_getnft == 'yes' :
                getnft_fn()
            else :
                messagebox.showinfo('Cryptonite : Sell NFT', 'The
Selling was cancelled.')
        elif nft_status == 4 :
            if result[16] == 1 :
                nft_code = 1
                status = messagebox.askquestion('Cryptonite : Sell NFT
','Are you sure to Sell %s %i %s'%(nft_name, nftnear_price,'Near'))
                if status == 'yes' :
                    getthb = nftthb_price
                    status_payment = messagebox.askquestion('Cryptonite :
Confirm Selling', 'Confirm to Get %i %s'%(getthb, 'THB'))
                    if status_payment == 'yes' :
                        money = result[7] + nftthb_price
                        send_nft = result[16] - nft_code
                        sql = '''
                            UPDATE user_data
                            SET money=?, nft4=?
                            WHERE id=?
                        '''
                        cursor.execute(sql, [money, send_nft, select_user])
                        conn.commit()
                        update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Purchasing is Successfully.')
                        if update:
                            sql = "SELECT * FROM user_data WHERE username=? and password=?"
                            cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                            result = cursor.fetchone()
                            sellnft_fn()
                        if status_payment == 'no' :
                            messagebox.showinfo('Cryptonite : Sell NFT', 'The
Selling was cancelled.')
                        else :

```

```

    messagebox.showwarning('Cryptonite : Warnning', 'You do not
have this NFT\nyou need to Get NFT')
    status_getnft = messagebox.askquestion('Cryptonite : Get
NFT', 'Do you want to Get NFT?')
    if status_getnft == 'yes' :
        getnft_fn()
    else :
        messagebox.showinfo('Cryptonite : Sell NFT', 'The
Selling was cancelled.')
elif nft_status == 5 :
    if result[17] == 1 :
        nft_code = 1
        status = messagebox.askquestion('Cryptonite : Sell NFT
','Are you sure to Sell %s %i %s'%(nft_name, nftnear_price,'Near'))
        if status == 'yes' :
            getthb = nftthb_price
            status_payment = messagebox.askquestion('Cryptonite :
Confirm Selling', 'Confirm to Get %i %s'%(getthb, 'THB'))
            if status_payment == 'yes' :
                money = result[7] + nftthb_price
                send_nft = result[17] - nft_code
                sql = '''
                    UPDATE user_data
                    SET money=?, nft5=?
                    WHERE id=?
                    ...
                '''
                cursor.execute(sql, [money, send_nft, select_user])
                conn.commit()
                update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Purchasing is Successfully.')
                if update:
                    sql = "SELECT * FROM user_data WHERE username=? and password=?"
                    cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                    result = cursor.fetchone()
                    sellnft_fn()
                if status_payment == 'no' :
                    messagebox.showinfo('Cryptonite : Sell NFT', 'The
Selling was cancelled.')
                else :
                    messagebox.showwarning('Cryptonite : Warnning', 'You do not
have this NFT\nyou need to Get NFT')
                    status_getnft = messagebox.askquestion('Cryptonite : Get
NFT', 'Do you want to Get NFT?')
                    if status_getnft == 'yes' :

```

```

        getnft_fn()
    else :
        messagebox.showinfo('Cryptonite : Sell NFT', 'The
Selling was cancelled.')
    elif nft_status == 6 :
        if result[18] == 1 :
            nft_code = 1
            status = messagebox.askquestion('Cryptonite : Sell NFT
','Are you sure to Sell %s %i %s'%(nft_name, nftnear_price,'Near'))
            if status == 'yes' :
                getthb = nftthb_price
                status_payment = messagebox.askquestion('Cryptonite :
Confirm Selling', 'Confirm to Get %i %s'%(getthb, 'THB'))
                if status_payment == 'yes' :
                    money = result[7] + nftthb_price
                    send_nft = result[18] - nft_code
                    sql = '''
                        UPDATE user_data
                        SET money=? , nft6=?
                        WHERE id=?
                    '''
                    ...
                    cursor.execute(sql, [money, send_nft, select_user])
                    conn.commit()
                    update = messagebox.showinfo('Cryptonite :
Successfully', 'Your Purchasing is Successfully.')
                    if update:
                        sql = "SELECT * FROM user_data WHERE username=? and password=?"
                        cursor.execute(sql,[userinfo.get(),pwdinfo.get()])
                        result = cursor.fetchone()
                        sellnft_fn()
                        if status_payment == 'no' :
                            messagebox.showinfo('Cryptonite : Sell NFT', 'The
Selling was cancelled.')
                        else :
                            messagebox.showwarning('Cryptonite : Warnning', 'You do not
have this NFT\you need to Get NFT')
                            status_getnft = messagebox.askquestion('Cryptonite : Get
NFT', 'Do you want to Get NFT?')
                            if status_getnft == 'yes' :
                                getnft_fn()
                            else :
                                messagebox.showinfo('Cryptonite : Sell NFT', 'The
Selling was cancelled.')

```

```

def totalproperty_fn():
    global status_login, total_f
    if status_login == False :
        messagebox.showwarning('Cryptonite : You must login first.',
'You are not logged in, You must login first')
        backtologin()
    else:
        status_login = True
        global total_ftop,total_fbottom,total_ent, total_f

        total_f = Frame(root, bg="lightgreen")
        total_f.rowconfigure((0,1,2,3), weight=1)
        total_f.columnconfigure((0,1,2,3), weight=1)
        total_f.place(x=0,y=0,width=w,height=h)

        total_ftop = Frame(total_f, bg="white")
        total_ftop.rowconfigure((0,1),weight=1)
        total_ftop.columnconfigure((0,1,2,3,4),weight=1)
        total_ftop.grid(row=0,column=0,columnspan=4,sticky='news')

        total_fcenter = LabelFrame(total_f,text='Total Property')
        total_fcenter.rowconfigure((0,1),weight=1)
        total_fcenter.columnconfigure((0,1,2,3,4,5,6,7),weight=1)
        total_fcenter.grid(row=1,column=0,rowspan=5,columnspan=4,sticky
='news', ipady=100)

        total_fbottom = Frame(total_f, bg="pink")
        bg = Label(total_fbottom,image=login_bg)
        bg.place(x=0,y=0,width=w,height=h)
        total_fbottom.rowconfigure((0,1),weight=1)
        total_fbottom.columnconfigure((0,1,2,3,4),weight=1)
        total_fbottom.grid(row=8,column=0,columnspan=4,sticky='news')

        if result_user:
            if result_user[6] == 1 :
                Label(total_ftop, image=doge_pf,
bg='white').grid(row=0, column=0, sticky='w')
            elif result_user[6] == 2 :
                Label(total_ftop, image=mr_pf, bg='white').grid(row=0,
column=0, sticky='w')
            elif result_user[6] == 3 :
                Label(total_ftop, image=alien_pf,
bg='white').grid(row=0, column=0, sticky='w')
                Label(total_ftop,text=fullname,bg='white',font='Impact 25
bold').grid(row=0,column=0,sticky='es')

```

```

        Label(total_ftop,text='Total Property',bg='white',font='Impact
25 bold', fg='#1A3C40').grid(row=0,column=2,columnspan=3,sticky='ews')

        Label(total_fcenter,text='THB',font='Impact 50 bold',
fg='#417D7A').grid(row=1,columnspan=7, sticky=N)

        Button(total_fbotttom,text='Back To
Profile',image=ico_return,compound=LEFT,command=profile_fn).grid(row=8,
column=0,sticky='w',ipadx=5,ipady=10)
        Button(total_fbotttom,image=ico_reload,compound=RIGHT,text='Refr
esh',command=totalproperty_fn).grid(row=8,column=8,sticky='w',ipadx=5,i
pady=10)
        near = 352.09
        nftthb1 = 200000 * near
        nftthb2 = 30701 * near
        nftthb3 = 24434 * near
        nftthb4 = 3200 * near
        nftthb5 = 6175 * near
        nftthb6 = 123456 * near
        total = result[7] + (result[8]*btc_coin) + (result[9]*eth_coin)
+ (result[10]*doge_coin) + (result[11]*kub_coin) +
(result[12]*bnb_coin) + (result[13]*nftthb1) + (result[14]*nftthb2) +
(result[15]*nftthb3) + (result[16]*nftthb4) + (result[17]*nftthb5) +
(result[18]*nftthb6) + (result[19]*near)
        #total_ent.insert(0,total)
        Label(total_fcenter, text=total, font='Impact 80 bold',
fg='#1A3C40').grid(row=0,column=0,columnspan=7)

def help_fn():
    global help_f, status_help
    status_help = True
    help_f = Frame(root, bg="yellow")
    help_f.rowconfigure((0,1,2,3), weight=1)
    help_f.columnconfigure((0,1,2,3), weight=1)
    help_f.place(x=0, y=0, width=w, height=h)
    bg = Label(help_f,image=lobby_bg)
    bg.place(x=0,y=0,width=w,height=h)
    Label(help_f, image=img_help).grid(row=1, column=1, rowspan=2,
columnspan=2, sticky='news')

def exit_fn():
    exit = messagebox.askquestion('Cryptonite : Exit Program', 'Do you
want to Exit This Program?', default='no')
    if exit == 'yes' :
        root.destroy()

```

```

def retrivedata() :
    sql = "select * from user_data"
    cursor.execute(sql)
    result = cursor.fetchall()
    print("Total row = ",len(result))
    for i,data in enumerate(result) :
        print("Row#",i+1,data)

def backtologin():
    global status_login, result
    if status_regis == True and status_login == False:
        regis_f.destroy()
        root.title("Cryptonite : Login & Register")
        root.option_add("*font", "Verdana 16 bold")
        registerlogin_fn()
        userentry.delete(0,END)
        pwdentry.delete(0,END)
        userentry.focus_force()
    if status_help == False and status_login == False:
        root.title("Cryptonite : Login & Register")
        root.option_add("*font", "Verdana 16 bold")
        registerlogin_fn()
        userentry.focus_force()
    if status_help == True and status_login == False:
        help_f.destroy()
        root.title("Cryptonite : Login & Register")
        root.option_add("*font", "Verdana 16 bold")
        registerlogin_fn()
        userentry.focus_force()
    if status_login == True :
        exit = messagebox.askquestion('Cryptonite : Log Out', 'Do you
want to Log Out ?', default='no')
        if exit == 'yes' :
            result = ''
            root.title("Cryptonite : Login & Register")
            root.option_add("*font", "Verdana 16 bold")
            status_login = False
            registerlogin_fn()
            userentry.delete(0,END)
            pwdentry.delete(0,END)
            userentry.focus_force()

#Status for Check
status_login = False
status_help = False
status_regis = False

```

```

#Program resolution
w = 1300
h = 700

createconnection()
root = mainwindow()
#Import Image
img_login = PhotoImage(file='image/mr_profile.png')
main_img = PhotoImage(file='image/Cryptonite_logo.png').subsample(3,3)
alien_pf = PhotoImage(file='image/Alien_profile.png').subsample(3,3)
doge_pf = PhotoImage(file='image/Doge_profile.png').subsample(3,3)
login_bg = PhotoImage(file='image/login_bg.png')
lobby_bg = PhotoImage(file='image/lobby_bg.png')
mr_pf = PhotoImage(file='image/mr_profile.png').subsample(3,3)
img_doge = PhotoImage(file='image/doge.png').subsample(3,3)
img_eth = PhotoImage(file='image/ETH.png').subsample(3,3)
img_bitkub = PhotoImage(file='image/bitkub.png').subsample(3,3)
img_bitcoin = PhotoImage(file='image/bitcoin.png').subsample(3,3)
img_bnb = PhotoImage(file='image/BNB.png').subsample(3,3)
img_mona = PhotoImage(file='image/NFT_Monalisa.png')
img_nft3D = PhotoImage(file='image/NFT_3DBoy.png')
img_heath = PhotoImage(file='image/NFT_Heathead.png')
img_anti = PhotoImage(file='image/NFT_Anticoding.png')
img_lotus = PhotoImage(file='image/NFT_Lotus.png')
img_bored = PhotoImage(file='image/NFT_BoredM.png')
img_soldmonalisa = PhotoImage(file='image/NFT_MonalisaSold.png')
img_sold3d = PhotoImage(file='image/NFT_3dBoySold.png')
img_soldheat = PhotoImage(file='image/NFT_HeatheadSold.png')
img_soldanti = PhotoImage(file='image/NFT_AntiSold.png')
img_soldlotus = PhotoImage(file='image/NFT_LotusSold.png')
img_soldboredm = PhotoImage(file='image/NFT_BoredMSold.png')
img_help = PhotoImage(file='image/help.png')
img_blurmonalisa = PhotoImage(file='image/NFT_MonalisaBlur.png')
img.blur3dboy = PhotoImage(file='image/NFT_3DBoyBlur.png')
img.blurheathead = PhotoImage(file='image/NFT_HeatheadBlur.png')
img.bluranti = PhotoImage(file='image/NFT_AnticodingBlur.png')
img.blurboredm = PhotoImage(file='image/NFT_BoredMBLUR.png')
img.blurlotus = PhotoImage(file='image/NFT_LotusBlur.png')
root.iconphoto(False, main_img)
#Import Icon
ico_regis = PhotoImage(file='icon/regis.png').subsample(9,9)
ico_return = PhotoImage(file='icon/return.png').subsample(15,15)
ico_search = PhotoImage(file='icon/search.png').subsample(15,15)
ico_reload = PhotoImage(file='icon/reload.png').subsample(15,15)
#Import Button
btn_regis = PhotoImage(file='button/button_register.png')

```

```
btn_login = PhotoImage(file='button/button_login.png')
btn_back = PhotoImage(file='button/button_back.png')
btn_addcredit = PhotoImage(file='button/button_add-credit.png')
btn_marketcap = PhotoImage(file='button/button_market-cap.png')
btn_totalppt = PhotoImage(file='button/button_total-property.png')
btn_getcoin = PhotoImage(file='button/button_get-coin.png')
btn_getnft = PhotoImage(file='button/button_get-nft.png')
btn_sellcoin = PhotoImage(file='button/button_sell-coin.png')
btn_sellnft = PhotoImage(file='button/button_sell-nft.png')
btn_confirmpurchase = PhotoImage(file='button/button_confirm-
purchase.png')
btn_seenft = PhotoImage(file='button/button_see-nft.png')
btn_getnear = PhotoImage(file='button/button_get-near.png')

#Spy's Job
userinfo = StringVar()
pwdinfo = StringVar()
fname = StringVar()
lname = StringVar()
phonenum = StringVar()
user_name = StringVar()
pwdinfo = StringVar()
cfpwdinfo = StringVar()
avatarinfo = StringVar()
mycoinopinfo = StringVar()
amt_addcredit = StringVar()
amt_entinfo = StringVar()
spincoin_info = IntVar()
findoption = StringVar()
spincoinsell_info = IntVar()
getnearspninfo = IntVar()

registerlogin_fn()
root.mainloop()
```

ການພັດງານ ແລະ Source Code (Cryptonite.py)

Module

pip install requests

API

<https://api.bitkub.com/api/market/ticker>

(API_HOST = "https://api.bitkub.com")