國立中興大學資訊管理學系

碩士學位論文

利用智能合約實現單車共享經濟之研究

The research on Bike Sharing Economy Using Smart Contract

指導教授：林詠章　Iuon-Chang Lin

研 究 生：廖子淳　Tzu-Chun Liao

中 華 民 國 一 O 六 年 六 月

國立中興大學資訊管理學系

碩士學位論文

題目：利用智能合約實現單車共享經濟之研究

姓名：廖子淳　　學號：7104029051

經　口　試　通　過　特　此　證　明

論文指導教授　林詠章

論文考試委員　黃慧凰

李服文

林詠章

中華民國　106　年　6　月　27　日

摘要

近年來環保意識抬頭，越來越多的節能減碳措施備受各國政府重視與推廣，人們也開始逐漸改變生活習慣，響應綠色樂活的概念。其中，結合觀光旅遊與健康運動的自行車租借服務，一直都是一項很熱門的大眾交通運輸工具。在台灣，Youbike 是家喻戶曉的單車租借服務品牌，然而在中國，卻開始出現另一種「無樁共享單車」旋風，也就是沒有固定租借站點的一種單車租借方式。

除了了解兩者之間的差異與問題以外，本篇研究希望透過區塊鏈技術的智能合約創新方式，解決目前單車租借服務所遭遇的困難，並為實現單車共享經濟提出一個可行解。

關鍵字：區塊鏈、智能合約、共享經濟、單車租借

Abstract

People's awareness of environment has rise in recent years, more and more energy saving and carbon reduction policy were popularized and got attention by the government, people were also gradually changing their habits and customs to rejoin the concept of green life style. Bike rental service combine sightseeing with exercise, is always a popular public transport. In Taiwan, Youbike is a well-known brand of public bike rental service, however, in China, they have another type of bike rental service, which is a fully station-less bicycle-sharing system.

Except to find out the difference between the two type of bike rental service and each of their problems, this research also want to solve current bike rental service encountered difficulties by using blockchain technology and smart contract, and to make a feasible solution for the realization of bicycle sharing economy.

Keywords: blockchain, smart contract, sharing economy, bike rental

目錄

摘要 i
Abstract ii
目錄 iii
圖目錄 vi
第一章 緒論 1
1.1 共享經濟介紹 1
1.2 共享單車介紹 2
1.3 國內外單車租借服務之市場介紹 3
1.4 共享單車遭遇的問題 8
第二章 文獻探討 10
2.1 區塊鏈介紹 10
2.1.1 區塊鏈如何運作 10
2.1.2 區塊鏈的結構 11
2.1.3 如何取得共識 11
2.1.4 區塊鏈的種類 13
2.1.5 區塊鏈的應用 14
2.2 以太坊介紹 15
2.2.1 以太坊的特色 15
2.2.2 以太坊版本整理 16
2.3 智能合約介紹 16
2.3.1 智能合約如何運作 17
2.3.2 部署智能合約 18
2.3.3 智能合約的應用 19
2.4 區塊鏈與智能合約的問題 19
2.4.1 區塊鏈的安全性議題 20
2.4.1.1 51%攻擊 20
2.4.1.2 分岔問題 20
2.4.1.3 區塊鏈的規模大小 23
2.4.1.4 區塊鏈交易的確認時間長短 23

2.4.1.5 現行法規問題 24
2.4.1.6 整合既有系統的成本問題 24
2.4.2 智能合約的問題 24
2.4.2.1 安全議題 24
2.4.2.2 隱私議題 24
2.4.2.3 情境議題 24
第三章 研究方法 26
3.1 現有問題說明 26
3.1.1 用戶違規性問題 26
3.1.2 租車問題 26
3.1.3 車輛調度問題 26
3.1.4 車輛品質問題 27
3.2 智能合約單車租借 27
3.2.1 特色介紹 27
3.2.2 功能介紹 28
3.3 智能合約單車租借之服務流程 30
3.3.1 車輛註冊 30
3.3.2 車輛租借 31
3.3.3 車輛歸還 32
第四章 系統建置與評估 33
4.1 以太坊錢包 33
4.1.1 安裝以太坊錢包 33
4.1.2 介面與功能 35
4.2 撰寫合約 40
4.2.1 車輛註冊功能 40
4.2.2 合約初始化 41
4.2.3 合約主要功能 42
4.3 部署合約 45
4.4 執行合約 47
第五章 結論與建議 50

5.1 智能合約單車租借的優勢 50
5.2 智能合約單車租借的劣勢 50
5.3 結語 52
參考文獻 53

圖目錄

圖 1- 1：Airbnb.....2
圖 1- 2：Uber.....2
圖 1- 3：YouBike 微笑單車.....3
圖 1- 4：ofo bike.....4
圖 1- 5：Mobike 摩拜單車.....4
圖 1- 6：Bluegogo 小藍單車.....5
圖 1- 7：oBike.....5
圖 1- 8：Citi Bike.....5
圖 1- 9：Santander Cycles.....6
圖 1- 10：Velib.....6
圖 1- 11：Call a Bike.....7
圖 1- 12：各自行車租借系統之比較.....7

圖 2- 1：區塊之結構.....11
圖 2- 2：公有鏈.....13
圖 2- 3：聯盟鏈.....13
圖 2- 4：私有鏈.....14
圖 2- 5：以太坊的歷史版本.....16
圖 2- 6：區塊鏈與智能合約.....17
圖 2- 7：智能合約的區塊結構與內容.....18
圖 2- 8：硬分岔.....21
圖 2- 9：硬分岔的發生.....22
圖 2- 10：軟分岔.....22
圖 2- 11：軟分岔的發生.....23

圖 3- 1：智能合約單車租借.....28
圖 3- 2：智能合約單車租借的功能說明.....29
圖 3- 3：車輛註冊的服務流程.....30
圖 3- 4：車輛租借的服務流程.....31
圖 3- 5：車輛歸還的服務流程.....32
圖 3- 6：押金補償的服務流程.....32

圖 4- 1：安裝以太坊錢包(1).....33
圖 4- 2：安裝以太坊錢包(2).....34
圖 4- 3：安裝以太坊錢包(3).....34
圖 4- 4：以太坊錢包的主頁面.....35

圖 4- 5：MAIN ACCOUNT 頁面.....35
圖 4- 6：NEW WALLET CONTRACT 頁面.....36
圖 4- 7：SEND 頁面(1).....36
圖 4- 8：SEND 頁面(2).....37
圖 4- 9：CONTRACTS 頁面.....37
圖 4- 10：DEPLOY NEW CONTRACT 頁面(1).....38
圖 4- 11：DEPLOY NEW CONTRACT 頁面(2).....38
圖 4- 12：WATCH CONTRACT 頁面.....39
圖 4- 13 ：WATCH TOKEN 頁面與 token 設計預覽.....39
圖 4- 14：智能合約單車租借的系統架構圖.....40
圖 4- 15：車輛資訊備註.....41
圖 4- 16：合約的狀態宣告.....42
圖 4- 17：合約的初始化.....42
圖 4- 18：function rent().....43
圖 4- 19：function inUse().....44
圖 4- 20：function rentingFee().....44
圖 4- 21：選擇欲部署的合約.....45
圖 4- 22：進行部署.....45
圖 4- 23：輸入密碼確認執行部署.....46
圖 4- 24：等待確認合約發布.....46
圖 4- 25：BIKE RENTING 頁面資訊.....47
圖 4- 26：執行 Rent 功能.....47
圖 4- 27：執行 In Use 功能.....48
圖 4- 28：執行 Renting Fee 功能.....48
圖 4- 29：執行合約的最後一步.....49

第一章 緒論

1.1 共享經濟介紹

共享經濟(The Sharing Economy)，簡單來說，其實就是一種提供租賃服務的商業模式。在現今網路發展迅速的時代，人們因為想要快速方便的使用某項產品或資源，但又無力或不願購買此項產品/資源，取而代之的是透過租借方式的一次性消費，在網路世界中完成線上交易，得到想要獲得的服務。共享經濟就本質上來說，或許的確只是單純提供產品/資源的租借，但與傳統的租借服務不同的是，由於網路媒介與其他科技的普及化，使得原本或許只有公司或企業才有能力提供的出租服務，變成是一種每個人都能貢獻其所擁有資源，並且因此得到利益，形成點對點的資源共用之商業模式。

共享經濟有弱化擁有權，強化使用權的效用。在共享經濟的模式下，其核心理念，是讓原先未被充分使用的閒置資源，在透過有償租借給他人的方式，能獲得更有效的利用，進而提高資源的整體利用效率。常見的共享經濟服務包括：交換住宿、汽車共享、單車共享等。

在共享經濟領域中，最著名的莫過於 Airbnb 以及 Uber 兩家新創企業，在這裡也將它們進行簡單的介紹。

- Airbnb

Airbnb，成立於 2008 年 8 月，其公司總部位於美國舊金山，是一個提供個人租借房宿的網站平台企業。透過註冊成為平台會員，使用者除了可以在上面看到住宿物件的資料以外，也能得知該房屋物件的擁有人，也就是房東的相關資訊，像是房東的個人檔案、先前房客的評價與回覆等，為房宿出租提供透明性與些許安全性。

看似便利的創新服務背後，其最大問題是法律與安全方面的疑慮。例如：當發生房客竊取屋內財物時，該如何處理房東的財物損失賠償問題? 又或者如果房客被房東飼養的狗咬傷時，相關的責任歸咎該怎麼釐清? 而平台底下房屋物件與其所有人，是否有符合各國當地法規所制定的相關管理條例，並取得合法營業執照，也是 Airbnb 在實現房屋共享經濟服務時，所需要面臨的議題與克服的挑戰。

- Uber

Uber，成立於 2009 年 3 月，其公司總部也是位於美國舊金山，是一個於手機 App 上，提供汽車租賃與共乘的共享經濟平台。使用者可以在註冊成為會員之後，綁定信用卡作為支付工具，並透過 Uber 進行叫車服務，抵達所欲前往之目的地；用戶也能申請成為 Uber 司機，開著自己所擁有的車輛提供載客服務，賺取報酬。

與 Airbnb 相比，Uber 在安全與法律議題所遭遇的困難，以及爆發的負面新聞，都來得更具爭議性。就 Uber 日前在台灣被交通部裁罰巨款且勒令停業的事件來看，Uber 被認為是以「創新」做為規避遵守政府法令的藉口，其底下的司機沒有勞健保，而其他相關的法律責任也並未有保障消費者(乘客)的明確規範。除此之外，Uber 在國外的爭議事件也不少，像是西班牙的計程車司機抗議 Uber 車輛沒有載客執照、商業用車執照和商業載客用車保險；印度發生司機性侵女乘客事件等等，都是 Uber 被當地政府禁止的原因之一。

圖 1- 1：Airbnb

圖 1- 2：Uber

1.2 共享單車介紹

單車租借的服務行之有年，而共享經濟的這種炫風也席捲了自行車市場。目前單車的租借服務大致上可以分為兩種：一種是於定點租還車的站點式租借，就像傳統的自行車出租店，或是在台灣廣被國人熟悉的 Ubike 服務，都是屬於固定站點的概念；而另一種則是沒有固定站點的租還車模式，也就是目前共享單車大多數都有一個共同特色 — 沒有固定的租借站點，採取的是”騎到哪，還到哪”的無站點式租借模式。

比起以往傳統的定點租借，需要前往指定的租還車據點，共享單車的營運方式可說是大大提升了便利性，但這種方式也會造成一些問題的發生，像是當找不到合法的停車格時，使用者可能隨意停放車輛引發亂象，又或是如果投入的單車數量不夠多，在車輛過於分散的情況下，可能很難尋獲可租借的單車。

除此之外，單車的租借類型也可以被分為兩種：一種是傳統的人力自行車，另一種則是裝載電池的電動自行車。隨著行動網路的建置普及化，不管是一般自行車或是電動自行車，在結合物聯網的概念之後，使用者騎乘自行車時將可以有更多不同於以往的騎乘體驗，從一開始的行動支付租借，到常見的地圖導航功能之外，也可以加入紀錄騎乘者的速度、騎乘習慣、感測心跳數等多元化服務，這些都將會是成為吸引使用者騎乘的元素，並且讓越來越多人，無論是使用其租借服務，亦或是提供自己的車輛讓他人租借，都能加入共享單車的行列，使得共享

單車的概念更加普及，也為日常的交通運輸方式搭出最後一哩路。下面我們將針對國內以及國外提供單車租借服務的市場業者，進行初步介紹與比較。

1.3 國內外單車租借服務之市場介紹

在台灣，撇開各地方傳統觀光性質的私營自行車租借業者不談，除了有已經營運多年的 Youbike 微笑單車之外，近日還有來自新加坡的共享單車業者 — oBike，進駐台灣市場。在國外方面，中國則已經成為一個共享單車蓬勃發展的區域，包含 ofobike、Mobike、Bluegogo 等，都是其代表性業者；而歐美地區，荷蘭被普遍認為是最早開始嘗試共享自行車模式的國家，更有著自行車王國的稱號，在西班牙、英國、丹麥、德國、法國等地也都有自行車租借服務的身影，美國也有屬於當地的共享單車系統。不過和中國相比，歐美等地的共享單車市場似乎沒有比中國來的熱絡，很多營運商其實都已經退出市場，而原因不外乎就是營運上的虧損或是相關資金的匱乏問題。

- Youbike

YouBike，微笑單車，是一項台北市政府與台灣捷安特自行車公司合作發起的台北市公共自行車租賃系統服務計畫，其目的是推廣民眾騎乘自行車，並鼓勵民眾使用低污染、低耗能的公共自行車作為短程接駁交通工具，減少及移轉私人機動車輛之持有及使用，以達改善都市道路交通擁擠、環境污染及能源損耗目的。

YouBike 的租借是採固定站點的方式，而付款方式則是持完成註冊後的電子票證(悠遊卡、一卡通），進行感應扣款；而如果使用者為單次租借，也可以透過信用卡申辦，完成租車服務。YouBike 目前的營運地區包括台北市、新北市、桃園市、新竹市、台中市、彰化縣，其租借費率也因各地方政府之補助政策而略有不同。YouBike 於 2012 年 8 月起動試營運，至今全台已建置的站點總數高達 1065 站，並在 2016 年 8 月突破一億人次的總騎乘次數。

圖 1- 3：YouBike 微笑單車

- Ofo

Ofo，創立於 2014 年，是一家位於北京的共享單車公司，號稱是全球第一個無站點式共享單車平台。ofo 發跡於校園，直至 2016 年 10 月，已在中國 22 座城市、200 多所校園，累積提供超過 4000 萬次共享單車騎乘服務，成為中國規模最大的校園交通代步解決方式。其單車特徵是一身黃色，故又有小黃車的俗稱。

ofo 提倡開放式的共享平台精神，鼓勵用戶提供分享自己擁有的單車，加入 ofo 的行列，並透過互聯網創新模式調動各地的單車數量，達到提升自行車使用效率的目的。ofo 冀望在未來能不生產自行車，而是只利用媒合用戶與單車的方式，實現共享單車無國界的代步需求。

圖 1- 4：ofo bike

- Mobike

Mobike，摩拜單車，成立於 2015 年 1 月，並於 2016 年 4 月在上海正式營運，是一家提供智能共享單車服務的新創公司。除了提供中國國內城市使用共享單車服務以外，Mobike 也在 2017 年 3 月成功進入新加坡市場。

Mobike 的特色是，其單車配有智能鎖，結合 GPS 定位功能與通訊模組，用戶可透過 APP 線上定位、預約和使用附近的 Mobike，騎乘到達目的地後，只需就近找尋合適的停放點即可還車，而不用停靠在固定的站點規還，並在上鎖之後完成電子支付扣款。目前 Mobike 的車種分為兩種：經典版和輕騎版，除了計費價格與車身設計不同之外，所有上線供租借的 Mobike 均設有車鈴、GPS 定位系統以及智能鎖功能。

圖 1- 5：Mobike 摩拜單車

- Bluegogo

Bluegogo，小藍單車，於 2016 年 11 月在中國深圳市推出並啟用，是由天津鹿鼎科技有限公司研發並運營的無站點式共享自行車品牌。和 ofo 以及 Mobike 相比，Bluegogo 雖然同樣也是提供共享單車之服務，但其品牌核心理念更強調用戶的騎乘體驗，表示自己的優勢在於「好騎、安全、快速解鎖」，在眾多共享自行車品牌中更被用戶譽為「擁有最佳的騎乘體驗」。

用戶可透過小藍單車 APP、支付寶和微信，掃描停放在路邊的小藍單車之車身 QR code 解鎖單車，並在結束騎乘後將單車擺放於合適的停靠點即可。小藍單車目前提供三款車種，分別是 bluegogo、bluegogo Pro 和 bluegogo Pro 2，三種車款均配有智能鎖與車鈴，不同之處在於兩款 Pro 等級車種具變速功能，且 Pro 2 之車身採碳纖維結構，所以最為輕量化。

圖 1- 6：Bluegogo 小藍單車

- oBike

oBike，是一家來自新加坡的共享單車新創公司，於 2017 年 1 月在新加坡正式上線，並於 2017 年 4 月登陸台灣，在花東地區開始正式營運。同樣也是無站點的隨租隨還方式，oBike 和 Mobike 十分相似，都具有地圖定位車輛、預約車輛、QR code 解鎖功能，但除此之外，oBike 也有一些不一樣的改良，像是記錄公里數、計算用戶減少排放的二氧化碳，以及消耗的卡路里數。

oBike 日前在台灣式營運期間，引發的用戶違停單車爭議事件，是共享單車模式很常見的問題。在這部分 oBike 也希望透過系統中的用戶積分管理制度，規範騎乘者的行為，避免發生破壞單車或違法停放單車的亂象。

圖 1- 7：oBike

- Citi Bike

Citi Bike 是一套位於美國紐約的公共自行車系統，在 2013 年 5 月正式營運，是美國最大的公共自行車租借系統。有別於台北的 YouBike 營運是政府委託民間企業來建置自行車租借系統與硬體設備，但還是由政府來經營；紐約的 Citi Bike 完全沒有接受政府資金補助，而是全由民間企業出資和收取會員費用來營運。

由於是私營性質，所以 Citi Bike 是以營利為主要考量的公共自行車系統，其計費方式對非年租會員的用戶來說並不友善，且租借的實際操作流程也繁瑣不便，營運期間可說是風波不斷，許多設計都為人詬病，甚至因為營運範圍都以較高檔的區域為主，而被紐約市其他地區的居民譏為「有錢人獨享的腳踏車共騎系統」。Citi Bike 總公司在 2015 年更換新的經營團隊，除了改善使用者的用戶體驗以外，也陸續增加新的站點與車輛，希望解決營運虧損的問題。

圖 1- 8：Citi Bike

- Santander Cycles

Santander Cycles，又稱作 Barclays Cycle Hire(BCH)，是一套位於英國倫敦的公共自行車租借系統，在 2010 年 7 月開始營運，屬於有固定站點式的租借模式。用戶可以使用銀行發行的信用卡付款，其費用則包括單車的租借費，以及騎乘費，並透過租借站機台所給予的密碼解鎖自行車。

由於在英國騎乘單車是需要有全套安全裝備(頭盔、護腕、護膝)才能上路，否則會被取締開罰，一般遊客通常都無法符合這項規定，而當地居民又普遍習慣使用自己既有的自行車與裝備，所以使用 Santander Cycles 的人並沒有很多。

圖 1- 9：Santander Cycles

- Velib

Velib 是一套位於法國巴黎的公共自行車租借系統，於 2007 年 7 月開始營運，也是屬於有固定站點式的租借模式，透過與世界排名第二的國際性戶外媒體廣告公司—德高集團（JCDecaux）合作，發展至今已成為歐美國家中最大的單車租借系統，目前已擁有 1750 個站點和 23600 輛自行車。

和紐約、倫敦等地相比，Velib 收取的租借年費幾乎是全球各大城市當中最低廉的，只需支付 29 歐元的年租金，便可以無限次使用公共自行車。這或許也是為什麼 Velib 擁有十分可觀的市場滲透力與獲利能力，讓當地即便是非中產階級的民眾也能負擔並願意使用該服務，同時也積極經營外國觀光客群。

圖 1- 10：Velib

- Call a Bike

位於德國各大城市的 Call a Bike，是一套由德國鐵路公司 Deutsche Bahn（DB，如同台灣的台鐵）所經營的公共自行車租借系統，早在 2000 年便已開始營運。Call a Bike 屬於無站點式的租借模式，用戶透過手機 APP 上的 GPS 定位功能找尋車輛，並輸入欲騎乘的車輛編號，獲得一組密碼後便可解鎖騎車，並在結束騎乘後將單車停放在合適之停車點即可。

Call a Bike 在早期所使用的租借服務方式，需要上網事先註冊資料，再透過電話撥打至客服中心要求車輛租借，並提供密碼解鎖單車，對於有語言隔閡的國外旅客而言並不方便，但現在推行的行動裝置 APP 版本，則有望改善這個缺陷。

圖 1- 11：Call a Bike

除了上述所介紹的單車系統以外，其實世界各地還有許多屬於自己國家的自行車租借系統，像是丹麥哥本哈根的 Donkey Republic、荷蘭阿姆斯特丹的 Urbee、西班牙巴塞隆納的 Dropbyke、澳洲墨爾本的 MBS、韓國首爾的 Ddareungi、香港的 GoBee.Bike、印度德里的 Greenolution 等，儘管大部份的營運模式還是採取固定站點的方式租還車，但無論哪一種，卻也都是單車共享經濟的實現。圖 1-12 為針對上面所提到的 9 個公共自行車租借系統進行整理與比較。

	Youbike	ofo	Mobike	Bluegogo	oBike	Citi Bike	Santander Cycles	Velib	Call a Bike
國家	台灣	中國	中國	中國	新加坡	美國	英國	法國	德國
有無固定站點	有	無	無	無	無	有	有	有	無
租借方式	靠卡感應解鎖	手動解鎖密碼鎖	QR code 掃描解鎖	QR code 掃描解鎖	QR code 掃描解鎖	手動輸入密碼	手動輸入密碼	手動輸入密碼	手動輸入密碼
付款方式	電子票證、信用卡	手機行動支付	手機行動支付	手機行動支付	信用卡、金融卡	信用卡	信用卡	信用卡	信用卡
有無押金制度	無	有	有	有	有	有	有	有	有

圖 1- 12：各自行車租借系統之比較

1.4 共享單車遭遇的問題

從上一個章節的討論，我們可以發現幾個特點，那就是中國的共享單車之營運模式，都是採取無固定站點的租借方式，提供使用者隨租隨還的騎乘服務；而其他國家目前則還是以固定站點的租借方式提供服務，並開始陸續出現仿效中國的無站點營運模式。另一方面，從單車租借的付款方式來看，也可以發現中國因為第三方支付的蓬勃發展，使用手機進行行動支付司空見慣，這點在共享單車的服務上也不例外，各大業者均採用行動支付的付款機制；而歐美等地則是仍透過較傳統的信用卡消費方式，作為單車租借時的交易手段。

不論是有無固定站點的租借型態，當中都會有各自的優勢與劣勢，不管是在租還單車時的流程，或是付款時採取的方式，勢必會面臨一些挑戰與遭遇種種需要克服的問題。下面將針對固定站點與無站點的共享單車系統，探討其各自擁有的優缺點，並歸納出兩種現行共享單車系統所遭遇的問題。

- 固定站點式：

目前各國單車市場中最常見的一種，相較於無固定站點式的設計，發展較早，屬於傳統的單車租借型態。採取固定站點式的共享單車系統有：台灣的 Youbike、美國紐約的 Citi Bike、美國華盛頓的 Capital Bikeshare、英國倫敦的 Santander Cycles、法國巴黎的 Velib、澳洲墨爾本的 MBS、韓國首爾的 Ddareungi、印度德里的 Greenolution 等。

其優點包括：

1 管理方便，車輛的調度規劃較簡單
2 車輛品質較一致
3 車輛較不容易遭竊
4 不會產生車輛違停的亂象
5 租車費用較便宜

而缺點則有：

1 需要找尋設置站點的地方才能完成還車
2 可能發生站點沒有空位可停的情況
3 租借的前置作業通常較為繁瑣，需在各站點所設立的服務機台進行操作
4 需要耗費成本與額外的空間場域建置站點

- 無站點式：

由中國的 ofo 共享單車公司提出，並在中國蔚為風行的創新單車租借營運模式，結合共享經濟的概念，近年來在世界各地都開始有應用此型態的單車營運商出現。採取無站點式的共享單車系統有：中國的 ofo、Mobike、Bluegogo 以及永安行、新加坡的 oBike、德國的 call a bike、丹麥哥本哈根的 Donkey Republic、荷蘭阿姆斯特丹的 Urbee、西班牙巴塞隆納的 Dropbyke 等。

其優點包括：

1 可隨地還車，便利性高

2　解決站點沒有空位可停的情況
3　透過手機操作即可完成註冊與租車等各式服務，且流程簡單快速
4　省去建置站點的土地成本與硬體成本

而缺點則有：

1　營運商不易管理，車輛調度的費用高昂
2　若車輛來源為用戶提供，則車輛品質將參差不齊
3　可能發生單車被用戶私藏佔用的問題
4　容易發生車輛違停的亂象
5　若單車數量不夠多，用戶將很難找到可租借的車輛

我們可以從上面的比較發現，固定站點式與無站點式，其各自擁有的優勢和劣勢，其實是相對的，而無站點式所引發的問題，其原因往往來自於人性。無站點式的設計解決了固定站點式的車輛停放問題，但卻又造成了另一個單車違停或亂停的爭議；而無站點式雖然讓用戶在還車時更加方便，卻也可能因為還車地點偏僻，或用戶的不當使用，造成下一位租車者的不便，讓單車被繼續使用的機率降低，而這除了產生車輛調度的成本問題以外，也大大違背了共享經濟是讓資源能被更有效利用的初衷。綜合上述兩種營運系統的特點，我們可以歸納出目前單車共享經濟所遭遇的幾個困境：

- 如何解決車輛違停與亂停，或是針對破壞車輛等其他不當行為進行規範
- 如何在沒有設置站點的情況下，解決找不到可供租借的車輛之問題
- 如何減少用戶將車輛放置在相對偏遠處，並有效降低車輛調度成本之問題
- 如何在不自行生產單車，而是落實共享經濟之精神，讓用戶主動提供自身單車的前提下，解決車輛品質參差不齊之問題

以上四點，都是目前單車共享經濟在實際上線營運後，共同面臨的挑戰，而這也是本篇論文的研究目的，希望透過結合區塊鏈技術與智能合約的概念，能有效改善問題並提出解套方法。

第二章 文獻探討

2.1 區塊鏈介紹

區塊鏈是一項結合密碼學、數學、演算法與經濟模型的技術，利用 P2P 網路和分散式共識演算法解決傳統分散式資料庫同步的問題，是一種整合多領域的基礎設施。

區塊鏈技術包括以下六大要素

- 去中心化：去中心化是區塊鏈的基本特色，意味著區塊鏈不再需要依賴一個中心節點才能運作，資料可以分散式地被記錄、儲存以及更新。
- 透明性：區塊鏈系統上的資料紀錄對每一個節點都是透明公開的，當資料更新時亦是如此，這也是為什麼區塊鏈是能被信任的原因。
- 開源性：幾乎大部分的區塊鏈系統都是開放給所有人的，上面的紀錄可被公開查詢，而欲開發系統的使用者也能運用區塊鏈技術創造他們想要的各式應用。
- 自主性：基於共識原理，區塊鏈系統上的每個節點皆能安全地傳遞或更新資料，目的是為了讓彼此之間的信任能從原本的對單一個體或機構轉變為對整體系統的信任，這當中也將不受任何人干預。
- 不可篡改性：任何紀錄都將被永遠保存，也不能被更改，除非有人能同時掌控高達 51%以上的節點運算力。
- 匿名性：區塊鏈技術解決了節點與節點之間的信任問題，資料傳遞或進行交易時只須知道對方的區塊鏈位址，達到匿名的特性。

2.1.1 區塊鏈如何運作

區塊鏈主要的運作流程如下：

1. 負責傳遞的節點記錄新的資料，並廣播到全網。
2. 負責接收的節點檢查接收到的資料之訊息，如果訊息無誤則將資料儲存至區塊裡。
3. 系統中的所有接收節點對區塊執行共識演算法。
4. 當共識演算法完成運算後，區塊將被儲存進區塊鏈當中，系統裡的每個節點承認此區塊並從此區塊向後繼續延伸區塊鏈。

2.1.2 區塊鏈的結構

一般來說，區塊的區塊內容包括主要資料、上一區塊之 Hash 值、當前區塊之 Hash 值、時間戳記以及其他資料。圖 2-1 說明區塊之結構。

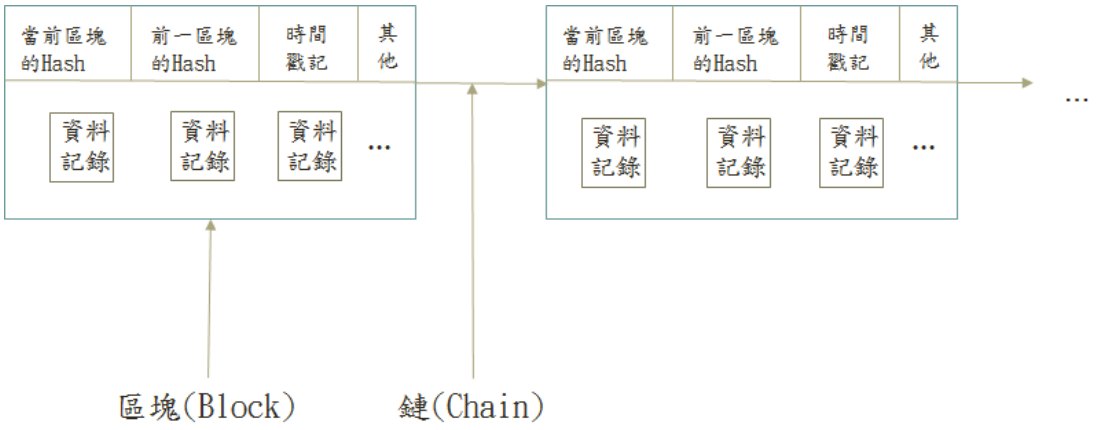


圖 2- 1：區塊之結構

- 資料紀錄：此部分的資料會根據這個區塊鏈是什麼樣的應用或服務而有不同，例如：交易紀錄、合約記錄、銀行清算記錄、產品履歷等。
- Hash 值：當一筆交易產生時，它將被加密為一組代碼並廣播到全網。由於每個節點的區塊中可能包含數百到數千筆的交易紀錄，區塊鏈利用 Merkle Tree 機制將那些大量交易紀錄結合並產生一最終 Hash 值，也就是所謂的 Merkle Tree Root。此最終 Hash 值將會被記錄在區塊的標頭中，成為當前區塊之 Hash 值。
- 時間戳記：紀錄區塊生成的時間。
- 其它資料：例如區塊的簽章、Nonce 值或其它使用者自行定義的資料。

2.1.3 如何取得共識

共識機制是一種使區塊鏈上的所有節點對相同訊息達成協議的方法，可以查驗最新的區塊正確地被加入區塊鏈當中，確保被節點儲存的資料為同一份訊息，並且防止區塊分岔的情況，甚至也能防止惡意攻擊發生。以下介紹兩種共識機制的原理，一種是 PoW 工作量證明機制，另一種為 PoS 權益證明機制。

- 工作量證明機制 (PoW, Proof of Work)

工作量證明機制，顧名思義即是工作量的證明，獲得證明即代表在過去曾經付出努力，比特幣就是採用工作量證明方法來管理系統，而就現實生活中的例子來說，獲得學位，或是考取執業證照等，都是工作證明的概念。

工作量證明機制的基本步驟如下：

1. 節點檢驗系統上的資料，將通過驗證的紀錄進行暫存。
2. 節點運用自身算力嘗試計算不同的隨機值(Nonce)，直到找出條件相符的隨機值。

3. 找出隨機值後會產生區塊，填入標頭資料後將資料紀錄在新區塊當中。
4. 接著廣播出新生成的區塊，通過其他節點驗證之後，把此新區塊加入原有區塊鏈上，然後所有節點從此新區塊後面開始接續進行工作量證明與區塊生成的作業。

每個區塊都有一個隨機值稱做「Nonce」在其區塊標頭，當每個節點進行工作量證明計算時，目的便是要藉由更動此 Nonce 值，使得區塊標頭之 Hash 值，小於一個預設好的「難度目標值」。其中，難度大小所代表的是當節點在計算 Hash 值小於目標值時所需要耗費的時間，也可以說是產生一新區塊所需要的時間。

在步驟 2 中，節點需要不斷消耗算力(computing power)與電力進行運算，經過大量嘗試計算(trial and error)後，找出符合的隨機值。這個計算隨機值的過程也就被稱做為「挖礦(mining) 」 ，而進行驗證的節點則稱為「礦工(miner) 」。

挖礦像是為區塊鏈系統提供了誘因與獎勵，其主要功能有三者：

1. 發行新的貨幣，挖礦成功的礦工獲得一定的報酬。
2. 調整區塊生成的時間，維護系統的支付功能。
3. 透過算力機制確保系統的安全性。

然而工作量證明機制存在著兩個缺陷，第一個是算力的耗費，第二個則是算力集中的問題。在挖礦的過程中，尋找隨機值所消耗的算力、電力或其他資源成本實在太高，除了成功挖礦的節點能有回饋，其他節點所投入的資源則都將形成一種浪費。而算力集中是一個安全性的問題，為了能順力挖到區塊，使用者將算力集中起來聯合挖礦，形成所謂的「礦池(mining pools) 」，這種情形將可能導致壟斷而產生疑慮。

- 權益證明機制 (PoS, Proof of Stake)

相對於工作量證明機制，需要耗費大量運算資源與成本，權益證明機制則不需要昂貴的電腦硬體成本或運算能力。權益證明機制的概念是：成功挖到新區塊並獲得報酬的方式，是取決於參與者在系統中所佔有的權益比例多寡，而不再是由挖礦的算力所決定。當一個使用者所持有的貨幣佔系統全部流通貨幣的 5%時，則該使用者能獲得新區塊的機率便是 5%。以太坊目前雖然是採用工作量證明機制運行以太坊系統，但它在未來預計會轉換為權益證明機制繼續提供服務。

當面對惡意攻擊時，權益證明機制在安全性上提供了一些額外的保護，之所以能有額外的保護，是基於下列兩個理由：

1. 執行攻擊所耗費的成本將會更加昂貴。
2. 攻擊後所能得到的好處將減少。

由於權益證明機制的原理是透過權益多寡進行運作，當攻擊者要發動攻擊時，首先他必須持有大量有價貨幣，才有可能成功影響系統，可想而知，事先

要能獲取多數貨幣是十分耗費成本的，再者，因為他擁有系統中大量貨幣，所以當成功發動攻擊時，攻擊者本身將是受到最大影響的人。

2.1.4 區塊鏈的種類

區塊鏈的種類依照使用者對系統的權限程度而有所不同，可以約略被劃分為下列三種型式

- 公有鏈：任何人都可以查看或驗證區塊鏈上的交易資料，也可以參與達成共識的過程，像是比特幣(Bitcoin)與以太坊(Ethereum)就是公有鏈的實現應用。圖 2-2 說明公有鏈的概念。

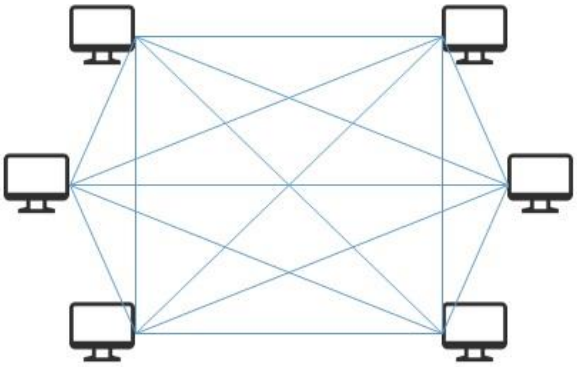


圖 2- 2：公有鏈

- 聯盟鏈：聯盟鏈的概念是指擁有權限的節點可以事先被篩選，而這些獲得授權的節點之間通常是有合作或商業關係的。聯盟鏈上的資料可以是公開或私有，這種概念可視為是部分去中心化的實現。像是 IBM 的 Hyperledger 計畫或是銀行之間的 R3CEV 聯盟都是聯盟鏈的一種。圖 2-3 說明聯盟鏈的概念。

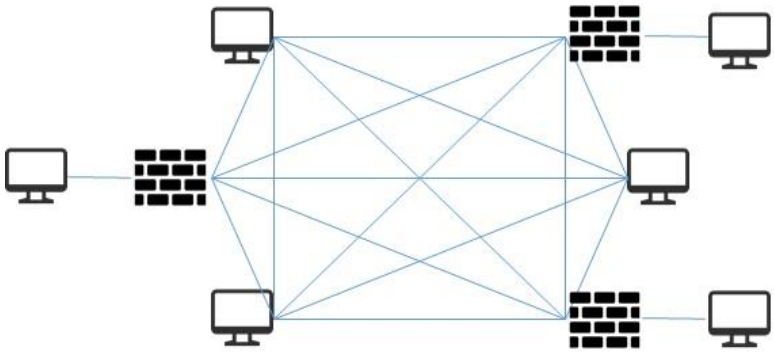


圖 2- 3：聯盟鏈

- 私有鏈：節點的權限將會被有所限制，並非每個節點都能參與此區塊鏈，資料存取的權限將會被嚴格的管理。圖 2-4 說明私有鏈的概念。

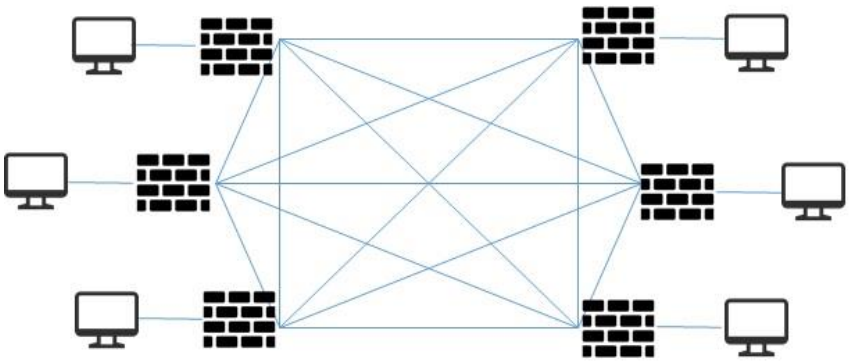


圖 2- 4：私有鏈

無論是哪種型式的區塊鏈，都將有各自的優勢與缺陷。有的時候我們需要使用公有鏈，因為它有最大的便利性；但也有情況是我們選擇聯盟鏈或私有鏈的方式，因為需要其隱私控制的特性，而這都將隨著使用者想要提供什麼樣的服務，或應用在何種場景，而有所不同。

2.1.5 區塊鏈的應用

不單只有金融方面的服務，區塊鏈技術也可以被應用在其他各種不同的領域，下面將簡介三項代表性的區塊鏈應用實例。

- 數位貨幣：比特幣（Bitcoin）

在 2009 年由開發者中本聰發行，比特幣的資料結構與交易系統均是利用區塊鏈技術打造，是一種數位貨幣與線上支付系統，並藉由加密技術實現去中心化的交易模式。比特幣透過公開金鑰位址傳遞與接收比特幣，並記錄其資料，完成匿名交易。而確認交易的過程則需要其它使用者的運算力來達成共識，最後再將交易紀錄在區塊鏈系統裡。

- 智能合約：以太坊（Ethereum）

智能合約是一種管理使用者之數位資產的數位合約，制定合約參與者的權利與義務，並且能藉由預先撰寫好的程式系統自動執行合約。智能合約本身不單單僅止是系統程序，也可以被視為是合約的其中一個參與者，會根據接收到的資料，做出相對應的回應，並儲存其資料，本身也能傳送訊息或值到外界。智能合約就像是一個可以被信任的第三者，暫時保管欲交易的資產，並遵循系統的程序指令執行合約內容。

以太坊，是一個結合智能合約的開源式區塊鏈平台，提供去中心化虛擬機器(EVM)處理合約的運行，並透過旗下的數位貨幣 – 以太幣（Ether），讓使用者能在此平台上開發各式不同的應用服務或智能合約。

- 超級帳本（Hyperledger）：

超級帳本是一個由 Linux 基金會在 2015 年 12 月創建的開源式區塊鏈平台，支援以區塊鏈技術為基礎的分散式帳本計畫。致力於設立一滿足全

球商業交易的帳本技術，其涵蓋範圍包括許多科技公司、金融機構或供應鏈公司，目標是希望整合並滿足不同行業的需求，為私有鏈應用制定開放的協議和標準。目前其五個孵化器項目之一的 Hyperledger Fabric 已經發布，其設計目的是做為分布式應用程式的基礎。

- 其它應用

目前還有很多區塊鏈技術的其它應用，像是智慧財產權的保護、供應鏈的產品追溯性、身分識別、保險、投票、跨境支付、物聯網、醫藥領域上的病患隱私管理以及博弈市場等，都是區塊鏈可以開發應用的領域。

2.2 以太坊介紹

以太坊(Ethereum) ，是一個有智能合約功能的開放區塊鏈平台，透過其底下的專屬加密貨幣以太幣(Ether)，提供去中心化的虛擬機器來運行，並具有圖靈完備(Turing Complete)特性，讓使用者可以透過此平台建立各式各樣的應用與服務。

以太坊最初在 2013 年由一位加密貨幣學家與程式設計師 Vitalik Buterin 發表，這個計畫在 2014 年 8 月透過預售以太幣，以網路群眾募資的方式得以開始發展，並在之後由一個非營利機構「以太坊基金會」(Ethereum Foundation)開發。

2.2.1 以太坊的特色

以太坊就好比 Android 系統，是一個開放度與開發性都很高的平台，它提供開發者一個開源的基礎系統架構，讓他們能在上面創建各種自己想要的應用或服務，並共同維護系統的完整性以及發展。和比特幣相比，在比特幣中需要花一小時確認一筆交易，但在以太坊中，因為區塊大小是動態調整的，一筆交易只需要 90 秒便能完成認證，速度相對提升很多，所以可以實現智能合約服務。

以太坊的特色包含下列幾點：

1. 以太幣(Ether)：以太坊的專屬加密貨幣，它的貨幣代號可以被簡化為 ETH，並可以跟其他加密貨幣進行交易，它也被用於使用以太坊服務與運算時，所需要給付的手續費。以太幣像其他實體貨幣的匯率一樣，都是有可能產生變動的，例如 The DAO 被駭客攻擊時，以太幣對美元的匯率從 $21.5 跌至 $15。
2. 智能合約(Smart Contract)：簡單來說，智能合約就是一種透過預先撰寫好的程式碼，提供服務並根據條件的不同，執行相對應程式功能的數位合約。智能合約的詳細說明我們將在下一個章節做解釋。
3. 叔塊(uncle block)：在比特幣系統中，當主要的母鏈發生分支，產生另一條較短的區塊，並且無法被併入母鏈之中，這條較短鏈上的區塊被稱之為

叔塊。但在以太坊系統裡，則鼓勵礦工將叔塊一併加入鏈結中。

4. 權益證明機制(Proof of Stake)：以太坊推崇權益證明機制的方式，節省使用工作量證明機制時所消耗的大量電腦運算資源，並在未來將會實作此機制的以太坊版本。
5. 閃電網路(lightning network)：閃電網路技術的目的是希望能夠提升交易速度，並且減少區塊鏈因資料量越來越大所帶來的負擔，進而提高其擴展性。目前以太坊尚未將此技術實作在當前的版本。
6. 硬分岔(hard fork)：以太坊使用硬分岔方式進行系統的改版，有關於硬分岔的說明會在 2.4 章節詳細介紹。

2.2.2 以太坊版本整理

從最初發佈的版本- 「Frontier」至今，以太坊歷經數次所謂的硬分岔改版，其中第三次分岔發生在 2016 年 6 月，以太坊上的一個去中心化自治組織"The DAO"被駭客入侵，市值五千萬美元的以太幣被竊取控制，為了解決此次事件所造成的損害，以太坊最後再度進行了一次硬分岔，作出一個向後不相容(backwards-incompatible)的改變，讓所有的以太幣回歸原處。圖 2-5 說明以太坊版本的歷史資料。

Version	Code name	Release date
1	Frontier	July 2014
1.1	Protocol Update 1	August 4th, 2015
2	Homestead	February 29th, 2016
3	The DAO	July 20th, 2016
4	Spurious Dragon	November 18th, 2016

圖 2- 5：以太坊的歷史版本

2.3 智能合約介紹

智能合約(Smart Contract)，是一種儲存於區塊鏈上，具有自身狀態的應用程式，可以協助、確認並執行合約，並透過多種具圖靈完備性的程式語言實作。智能合約承襲區塊鏈的概念，具有去中心化的特性，合約內容對所有人也都是公開透明的，而以太坊正是一個實現智能合約的技術平台，這也是以太坊最重要的功能之一。

智能合約就像是一個存在於以太坊系統中的中間者，自身擁有以太幣地址，當用戶對合約的地址發送一筆交易後，該合約就被觸發，並且根據交易中的其他條件，合約會執行自身的程式碼，最後回傳一個結果，而這個結果也可能是從合約的地址發出另外一筆交易。

現行的區塊鏈系統具有不同程度的能力來支援智能合約，圖 2-6 將它們進行了簡單的歸納：

	無智能合約的區塊鏈	具有智能合約的區塊鏈	具有圖靈完備性的智能合約區塊鏈
內容	分散式儲存	分散式處理：可處理預置的運算邏輯	分散式處理：可處理任何邏輯
舉例	Bitcoin（公開式） Litecoin（公開式） Multichain（私有式）	NXT（公開式）	Ethereum（公開式） Eris（私有式） Clearmatics（私有式）

圖 2- 6：區塊鏈與智能合約

2.3.1 智能合約如何運作

利用區塊鏈技術運行智能合約時，主要的概念可以被定義為下列步驟：

1. 合約的參與者將共同參與制定或審核其合約的內容與功能。
2. 在制定完成後，合約將透過 P2P 網路廣播並儲存在區塊鏈上。
3. 智能合約將在日後根據其訂定條件，自動履行合約的內容與功能。

步驟一：制定智能合約

a. 首先，使用者必須在區塊鏈系統上進行註冊，獲得自己的公鑰與私鑰；公鑰將作為自身的帳戶位址，而私鑰則是存取使用者帳戶的唯一權限。
b. 合約參與者共同制定合約，透過程式語言訂定各自的權利和義務，並透過各自的私鑰簽署合約，以確保合約的有效性與合法性。
c. 合約將會依據其中的承諾內容，被傳送至區塊鏈系統裡。

步驟二：與區塊鏈系統連結

a. 在區塊鏈系統上的每個節點都會收到該份合約，負責驗證工作的節點會儲存此合約，等待新一輪的共識機制產生，觸發對此合約的共識處理。
b. 當要進行共識機制時，驗證節點會把所有合約集合起來打包成一份合約集，計算該合約集的 Hash 值，並透過此 Hash 值產生一個區塊接著廣播到系統中。而其它的驗證節點將會把此合約集的 Hash 值，與自己持有的合約集 Hash 值作比較，並傳送一份自己認可的合約集給其他驗證節點。在經過一連串的節點發送與驗證工作之後，最終將得到對最新之合約集的共識。
c. 透過區塊的形式，將最新的合約集傳送至系統。

每個區塊包含下列資訊：當前區塊的 Hash 值、上一區塊的 Hash 值、獲得共識時的時間戳記、以及其他合約資料。圖 2-7 說明智能合約的區塊結構與內容。

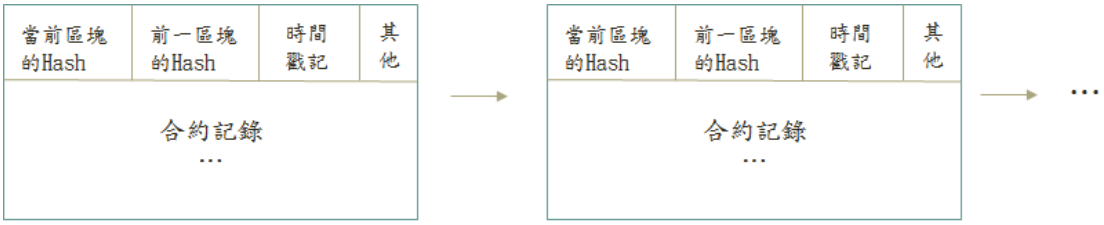


圖 2- 7：智能合約的區塊結構與內容

收到合約集的節點將會對合約上的協議逐項驗證，檢查像是參與者的私鑰簽章是否無誤等，如果通過驗證，最終合約才會被記錄在區塊鏈上。

步驟三：執行智能合約

a. 智能合約將會定期檢查其狀態或觸發之條件，並將符合條件的協議發送驗證工作，等待取得共識。

b. 進入最新驗證程序的合約協議，會被傳送到每個驗證節點驗證其簽章，通過驗證的合約協議內容才能進入等待共識的集合。等到其他多數節點也都通過驗證之後，該合約協議內容就會被成功執行並通知合約參與者。

c. 執行合約內容之後，智能合約將會再次檢查自身的合約狀態，當所有合約上的內容均完成執行之後，智能合約會把合約的狀態標記為已完成，並從最新的區塊中移除該合約。

2.3.2 部署智能合約

以下將針對佈署智能合約時，所需要了解的概念、工具、以及相關知識進行說明。

- 以太坊虛擬機器(Ethereum Virtual Machine，EVM)

  以太坊虛擬機器，或稱 EVM，是一個針對智能合約的應用，在以太坊上的運行環境。它不單只像是沙盒概念，而是完全地獨立運行，也就是說，在 EVM 中運行的程式碼不會受到外部網路或是其他系統的干預影響，甚至連智能合約彼此之間的存取也都是非常有限的。

- 程式語言

  智能合約可以透過數種不同的程式語言進行撰寫，在此份研究中，我們使用的是”Solidity”這個語言，因為它是目前所有可撰寫智能合約的程式語言當中，最受歡迎也最多人選擇使用的語言，除此之外，它也是以太坊官方推薦撰寫智能合約的程式語言。

  Solidity 是一個合約導向的高階程式語言，可以自行定義複雜類行，它的語法類似於 JavaScript，並且是針對於以太坊虛擬機器(EVM)進行設計。

- 編譯器

  使用“Solc”編譯器，是 Solidity 的命令行編譯器(solidity commandline compiler)。
- 客戶端軟體

  目前以太坊主要的客戶端軟體為”Geth”，而以太坊底下的以太坊錢包軟體(Ethereum Wallet)所使用的節點也是 Geth。Geth 可以被安裝在多種作業系統上，透過 Geth，使用者可以實現以太坊的各種功能，像是新增帳戶、開啟挖礦、交易以太幣、創建智能合約等。
- 去中心化應用程式(Decentralized Application，DApp)

  利用去中心化概念或技術，所開發出來的應用程式，便是去中心化應用程式。去中心化應用程式的概念類似於智能合約，但存在兩個主要不一樣的地方。首先，去中心化應用程式的參與者，不像智能合約通常只會有合約簽訂雙方，而是無數存在於市場中的人；其次，去中心化應用程式也不一定只具有交易功能的金融性服務，而是可以以各種類型存在於應用端。
- 佈署智能合約的步驟

  1. 開啟以太坊節點
  2. 透過編譯器編譯智能合約，得到一組二進制代碼
  3. 將編意好的合約佈署在網路上，得到該合約的區塊鏈位址
  4. 透過 web3.js 提供的 JavaScript API 來調整合約

2.3.3 智能合約的應用

智能合約的技術可以被應用在許多地方，主要的概念跟金融領域、保險業、以及租賃服務最為相關。例如：

- 房屋租賃
- 代幣系統
- 作物保險
- 證券登記與清算
- 彩券發行
- 音樂產業的智慧財產權

2.4 區塊鏈與智能合約的問題

儘管區塊鏈技術與其去中心化的概念已越來越被廣為人知，相關的應用與服務在國外更是蓬勃發展，而國內的金融機構和一些新創公司也對這部分在未來的展望躍躍欲試，但不可否認的是，如此創新的尖端技術雖然在很多領域都獲得關注，但它也存在著一些問題與挑戰是需要面對與克服的。

2.4.1 區塊鏈的安全性議題

以下將針對幾個區塊鏈技術可能會遭遇到的安全性問題，以及其他相關層面的隱憂進行研究與討論，其中包括：51%攻擊、分岔問題、區塊鏈的規模大小、區塊鏈交易的確認時間長短、現行法規問題以及整合既有系統的成本問題。

2.4.1.1 51%攻擊

透過工作量證明機制，挖到區塊的機率取決於礦工的工作付出(例如：耗費電腦的 CPU/GPU 不斷運算找尋目標值)，而由於此種機制的設計，使用者為了想要成功挖礦，可能會聯合起來共同挖礦，形成所謂的礦池。而一旦有礦池或是其他有心人士，持有高達 51%的區塊算力時，整個區塊鏈系統將可以被控制，而這也顯然產生了安全上的疑慮。

如果有人能擁有高達 51%，也就是過半的運算能力時，他/她將可以比其他用戶更快找到 Nonce 值，更可以決定哪個區塊是有效力的，當中能做到的事情有：

1. 修改交易紀錄，此動作可能進而造成雙重花費的攻擊(double-spending attack)。
2. 讓區塊停止驗證交易，進而癱瘓整個區塊鏈系統。
3. 不讓其他礦工挖出任何有效的區塊。

51%攻擊在過去因為大多數交易所包含的價值，遠大於挖到區塊的報酬獎勵而較具可行性，但現今隨著新的挖礦機制出現，51%攻擊發生的可能性相對於過去將來得較低。

2.4.1.2 分岔問題

分岔問題的發生，是因為當區塊鏈系統進行軟體更新時，其去中心化節點因版本不同，新舊節點對原本的共識協議與更新後的共識協議產生分歧，所引發的問題。區塊的分岔是一個很重要的議題，因為它跟整個區塊鏈系統息息相關，對整個區塊鏈系統更是影響深遠。

- 分岔的種類

當新版本的區塊鏈軟體發佈時，節點對共識規則裡的協議也發生改變，這時候，區塊鏈系統中的節點可以被分為兩種；遵循舊有共識規則的舊節點，以及接納更新後共識規則的新節點。此兩種節點可能會帶來以下四種情況：

1. 新節點接受舊節點所發出的交易區塊。
2. 新節點不接受舊節點所發出的交易區塊。
3. 舊節點接受新節點所發出的交易區塊。
4. 舊節點不接受新節點所發出的交易區塊。

由於有這四種在取得共識時所產生的不同情況，分岔問題因而發生，並且

根據這四種情況，分岔問題的種類可以被分為兩種：硬分岔以及軟分岔。在討論硬分岔與軟分岔之前，除了區分新舊節點的不同以外，還必須比較新舊節點之間的運算能力，而以下討論的情況，均假設在新節點之算力大於 50%的前提下進行說明。

- 硬分岔

硬分岔，是當區塊鏈系統因為有了新的版本或新的協議，並且與先前的版本不相符時，舊節點不接受新節點的挖礦結果，進而使原本只有單一一條的區塊鏈分裂為兩條。儘管新節點的算力大於舊節點的算力，舊節點仍然會繼續維護它認為是正確的那條鏈。圖 2-8 說明硬分岔問題。

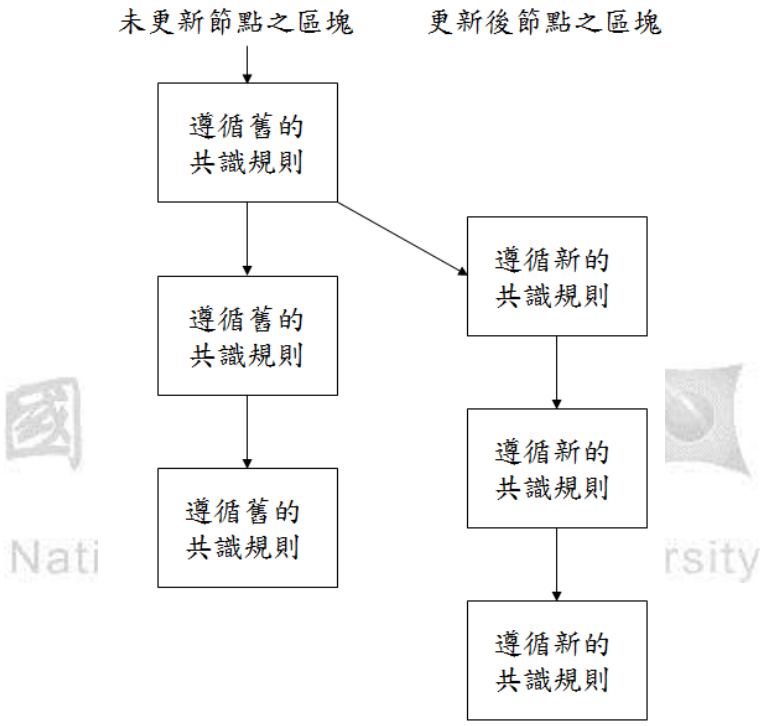


圖 2- 8：硬分岔

當發生硬分岔時，必須要求系統上的每個節點升級協議，而沒有升級的節點將無法像之前一樣正常地工作。如果系統存在更多未升級的舊節點，那它們將繼續在另一條不一樣的鏈上進行維護工作，也就是說原本既有的一條鏈將被分岔為兩條。圖 2-9 說明硬分岔會發生的原因。

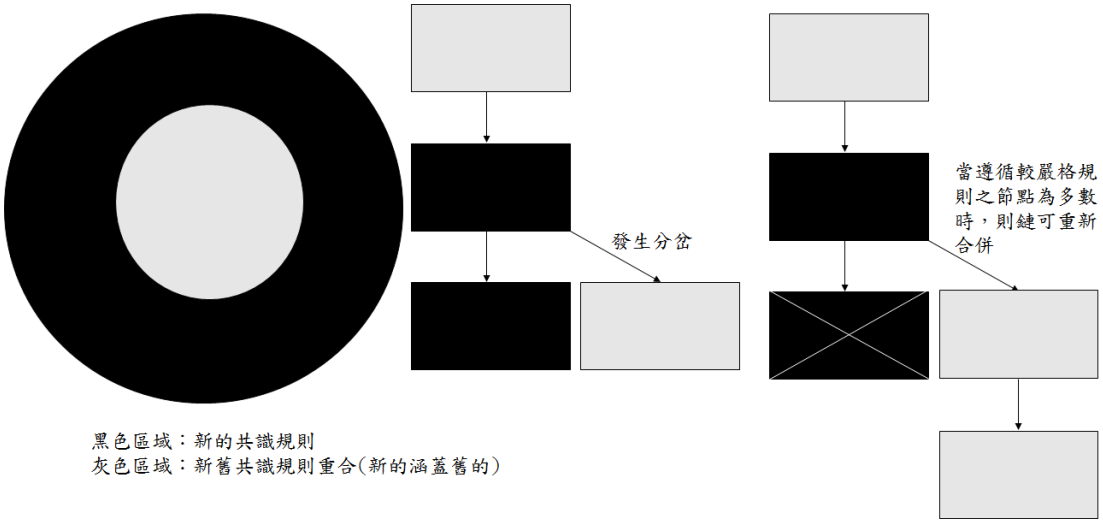


圖 2- 9：硬分岔的發生是因為舊節點的驗證條件比新節點來得更加嚴苛。

- 軟分岔

軟分岔，是當區塊鏈系統因為有了新的版本或新的協議，並且與先前的版本不相符時，新節點不接受舊節點的挖礦結果。儘管新節點的算力大於舊節點的算力，舊節點所挖出的區塊將永遠得不到新節點的認可，但新舊節點會繼續在原本同一條鏈上進行維護工作。圖 2-10 說明軟分岔問題。

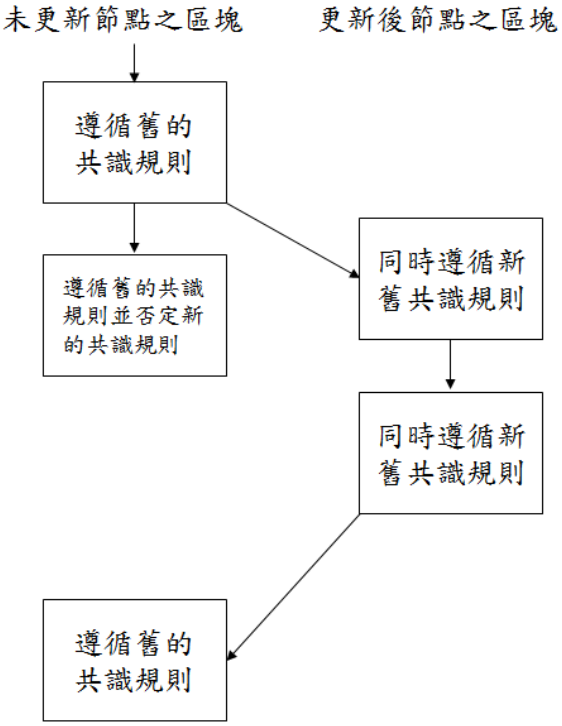


圖 2- 10：軟分岔

當發生軟分岔時，系統上的節點不需要同時升級新的協議，而是可以允許逐步升級。有別於硬分岔，軟分岔發生時只會有原本一條鏈，當節點升級時，軟分岔也不會對系統的穩定性與有效性帶來影響。儘管如此，軟分岔的情況讓舊節點無法察覺其實共識條件已發生改變，而這在某種程度上而言，是與區塊鏈中，每個節點均能有效進行驗證工作的原則相違背的。圖 2-11 說明軟分岔會發生的原因。

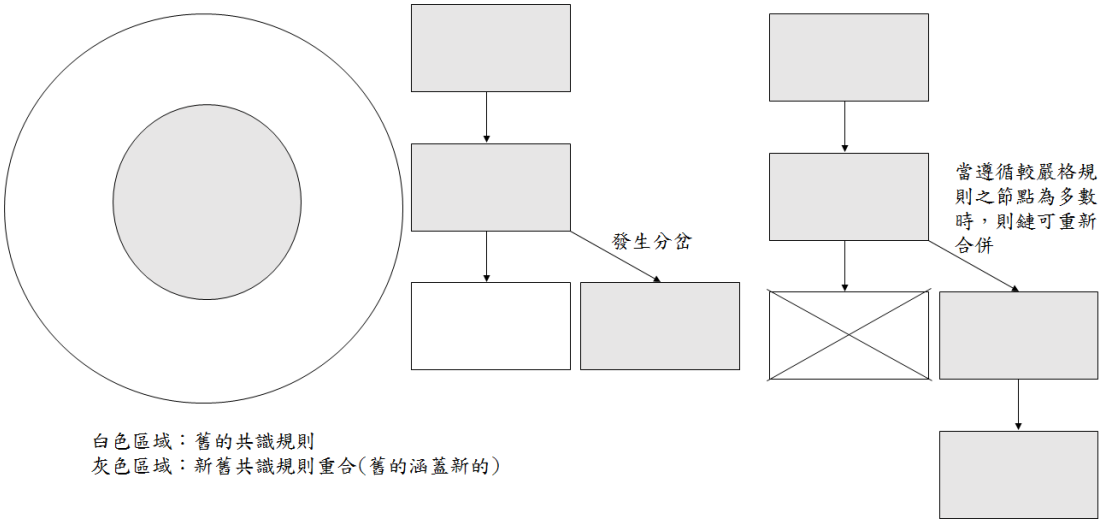


圖 2- 11：軟分岔的發生是因為新節點的驗證條件比舊節點來得更加嚴苛。

2.4.1.3 區塊鏈的規模大小

隨著區塊鏈的成長，上面記錄的資料也變得越來越多，其儲存與計算也變得越來越有負擔，而花費大量的時間在同步區塊資料的同時，又有新的區塊在繼續生成增加，讓用戶在運行系統時產生了很大的問題。簡易支付驗證技術為這個問題帶來了解套方式。

簡易支付驗證技術(Simplified Payment Verification ，SPV)，是一項支付驗證的技術，使用戶無須同步完整區塊鏈資料，而只需要使用區塊的標頭資訊即可。這個技術可以大大減少用戶在區塊鏈交易驗證上的資料量，降低用戶在交易量劇幅增加時所面對的系統承載壓力。

2.4.1.4 區塊鏈交易的確認時間長短

與傳統信用卡線上交易相比，有時可能需要花費二到三天才能完成確認一筆交易，比特幣只需要約莫一小時即可完成此驗證動作，雖然已經快速許多，但就理想的交易方式而言速度卻仍是不夠快。閃電網路則是一項可能實現這個需求的解決之道。

閃電網路(Lightning Network)，是一項利用 Hashed Timelock Contract (HTLC)以及 Revocable Sequence Maturity Contract(RSMC)兩種類型的交易合約所構成的技術應用，具有去中心化的特性，並透過雙向支付通道的方式，允許交易在多個 P2P 支付通道之間進行安全支付，而無須信任對方或第三方，形

成一個可以讓任何節點之間都能進行交易支付的網路。

2.4.1.5 現行法規問題

以比特幣做為例子，其系統去中心化的概念，將會使政府央行對經濟政策與貨幣數量的控管能力降低，這點讓政府對區塊鏈技術抱持謹慎的態度。相關機構必須盡快深入研究這項新議題，加速制定新的法規，才不會對市場產生風險，也確保法律在日後能保障我們的權益。

2.4.1.6 整合既有系統的成本問題

當一項新的技術，尤其是基礎設施，要對既有系統進行整合工作時，所耗費的時間與金錢成本都可說是十分龐大的，因為其牽涉的範圍與更動的領域都很廣大。我們除了確保區塊鏈技術可以為我們帶來經濟效益，並且符合監管條件以外，同時也必須與傳統組織介接，妥善處理來自組織內部的現有問題。

2.4.2 智能合約的問題

在探討完區塊鏈的問題之後，接著要談的是智能合約在可行性上面可能會遭遇的難題。結合區塊鏈的智能合約可說是一項十分新穎的技術，雖然仍並未大量使用在我們的日常生活中，處於剛起步的階段，但卻也已經有一些問題與挑戰存在其中。

2.4.2.1 安全議題

由於智能合約的環境是被設計為無須信任也能執行的，並且是去中心化的，因此，如果當進行交易時發生任何問題或是錯誤，將無法被修改。在目前的系統中，我們可以透過一些具管理權限的中央機構彌補過失，但在智能合約中，用戶則必須獨自承擔這種風險。

2.4.2.2 隱私議題

智能合約與區塊鏈一樣有著隱私性方面的問題，但又因為其本身所應用的底層技術而難以避免。任何在相同區塊鏈上的人都能看到合約的詳細資訊，通常對於簽訂合約的雙方來說，是非常公平公正的，但如果是簽訂的內容是一些商業貿易協定或者含有敏感資訊時，情況將變得複雜化，需要謹慎地處理合約的佈署方式，因為基本上來說，智能合約上的內容都將是透明公開的。

2.4.2.3 情境議題

智能合約看似是一個與人立訂承諾的完美工具，像是原本經濟條件較差，不能向銀行申請貸款的族群，能透過智能合約的方式與銀行簽訂契約，而銀行也能基於智能合約能不被干涉且自動履行的特性，不用擔心借款難以追討，替原本無法成為客戶的族群提供服務。話雖如此，讓我們來試想另一種情況：你

正駕駛著一輛透過智能合約租借的汽車行駛於高速公路上，突然之間，可能是因為租借交易沒有成功通過驗證，而遭到限制使用該車的操作權，這種情況將使得駕駛的生命安全受到威脅，因此，如何在正確的情境下適當地執行合約，也是一項非常需要謹慎處理的重要議題。

第三章 研究方法

3.1 現有問題說明

無站點式的單車租借服務實現了共享經濟在自行車產業的應用，也解決了原本固定站點式在還車時，可能會發生無位可停的情況，這種隨處都能停放單車的方式，為用戶帶來很大的便利性。儘管如此，這種便利性卻也引發了其他問題，以下我們將根據第一章末段所提到的內容，把現有單車共享經濟所遭遇的問題，分為四種類型：

一、用戶違規性問題

二、租車問題

三、車輛調度問題

四、車輛品質問題

3.1.1 用戶違規性問題

用戶違規性問題的項目有很多種，其中包括：用戶將單車私藏於自家，或其它可能導致別的用戶無法租借之地點、用戶將單車歸還在不適合停放的地點等違停/亂停之情況、用戶惡意損壞車輛、用戶偷竊車輛、用戶將單車鎖上私鎖、用戶騎乘時違反交通規則等，本研究將以上問題，均歸類於用戶違規性問題。

3.1.2 租車問題

從還車時的角度來看，無站點式的租借在這部分有很好的服務體驗，不用找尋指定的站點位置而隨意停放確實很方便，這點是固定站點所比不上的；但回到租車時的角度來看，這對下一個使用者而言，卻不一定是方便的，尤其是當車輛投放的數目不夠多，或是停放在偏僻地方的時候，這種情況下固定站點就顯示出其優勢所在。換句話說，無站點式的租借在某些情況下，只是將原本固定站點在還車時所碰到的不便性，轉移到找尋租借車輛的過程上而已。

3.1.3 車輛調度問題

無站點式的租借服務並非完全不需要調度車輛，有時可能因為營運需求或車輛維修等因素，需要將車輛載往特定地點進行後續處理。理論上來說，由於無站點式的租借允許單車不用停放在固定場域，因此，用戶可以將單車騎到任意想到達的地點再就近擺放即可，所以比起固定站點式的租借，車輛將更有機會被停放在偏遠地區或其他非租借密集性高的地區，此時就可能產生車輛調度的需求，以及運費成本問題。從這點看來，無站點式的租借服務所耗費的車輛調度成本，將比固定站點式來得高，而如何減少用戶將單車騎到對營運商而言，相對偏遠的地方之情形，進而降低營運成本，就顯得相當重要。

3.1.4 車輛品質問題

目前市面上提供租借服務的無站點式共享單車營運商，旗下的車輛來源幾乎都是與自行車品牌的車商合作，或是自行生產單車，來提供使用者騎乘。在這種情況下，由於是營運商統一生產，屬於商業營運用車輛，單車的車種與配備標準化，其車輛品質也較容易維持一定水準。但如果之後想要採取每個人提供自己的自行車，而不是營運商統一生產，作為共享單車方式的話，那麼勢必會面臨到車輛品質不一的問題。而在租借不同車種、車齡、車況的情形下，如何設計租借流程、制定公平的收費方法、規範使用方式等管理細則，對營運商而言，會是一大挑戰。

3.2 智能合約單車租借

綜合上述四點問題，本研究將結合區塊鏈技術與智能合約的應用，利用以太坊平台所提供之智能合約功能，建置一套智能合約單車租借方法，並希望透過此系統的實作，替現有單車共享經濟所遭遇的問題，提出有效的解決方法。

3.2.1 特色介紹

智能合約單車租借方法的設計適用於無站點式的營運模式，並希望能像共享經濟產業的兩大企業- Airbnb 與 Uber 一樣，將使用者手上擁有的閒置資源分享出去讓其他有需要的使用者使用，也就是媒合用戶(C 端)的”非標準化”資源，提供非標準化的服務體驗，形成有別於過往 B2C 的傳統思維，而是 C2C 的創新營運模式。

智能合約單車租借方法的概念是：讓用戶提供自己的單車，並透過以太坊平台撰寫具租借功能的客製化智能合約，使其他欲租借的需求方用戶在與單車擁有人簽訂完合約之後，把單車的使用權轉移到借方，完成租借服務，實現真正的單車共享願景。在這個方式中，任何人都可以將自己的單車透過簽訂智能合約的方式租借給他人，或是向他人租車。而根據智能合約的程式功能，用戶可以制定各種規定或條件，像是租借費用的金額、騎乘時間的限制等其他使用細則，創造各式各樣不同使用方式的單車合約。

這種租借方式有別於現有的營運模式，除了租借時的車輛來源是由用戶之間彼此互相提供，而非自行車車廠統一生產以外，租車時的對象也不再是面對大型企業，或是由他們集中管理，而是個人用戶之間進行交易與提供服務，形成點對點的租借模式。圖 3-1 說明智能合約單車租借方法的概念。

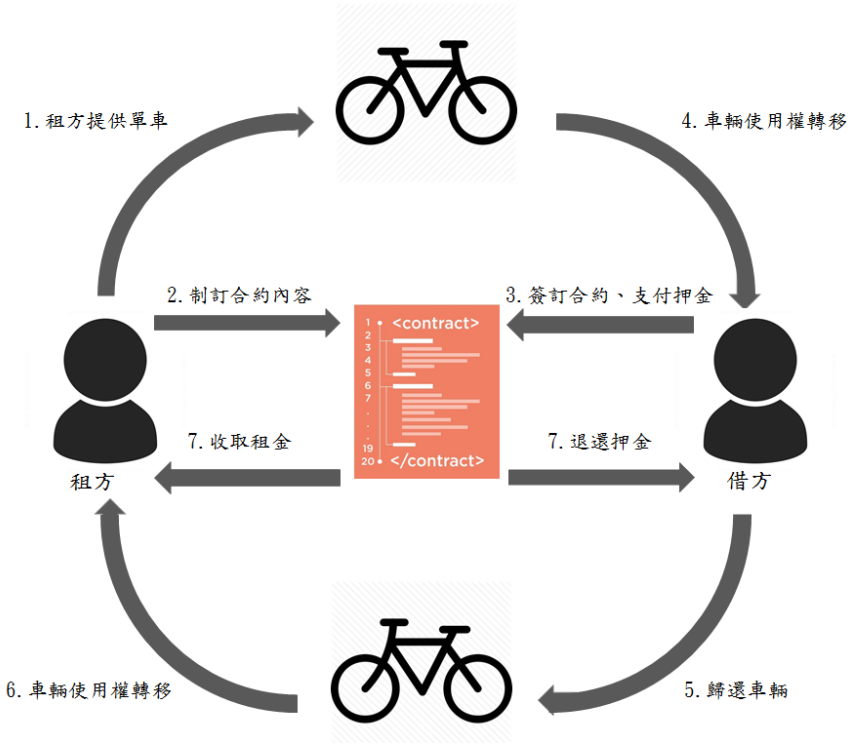


圖 3- 1：智能合約單車租借

3.2.2 功能介紹

為了克服上一小節所提到單車共享經濟所遭遇的問題，智能合約單車租借將針對此四種類型之問題設計一套對應的解決辦法，並利用智能合約撰寫相關功能，概念化呈現所提出的創新方式，其中包括：押金制度、積分制度、標示車輛使用程度與制定騎乘範圍。

- 押金制度：

押金制度是目前共享單車最常見的基本共同方式，而在本研究所提出的智能合約單車租借方法中，也是不可或缺的一環。押金制度的出現，讓傳統的「一份租賃商品對應一筆押金」方式，變成「一個用戶對應一筆押金」，也就是說同一輛車可能有多筆押金收取紀錄，對營運商而言，只要用戶數持續成長，那麼透過押金所收取的資金也會穩定增加。在智能合約單車租借方法中，押金制度也是一項保障車輛所有人的方式，當自己所提供的單車被偷竊或是破壞時，能有一定的補償保證措施。這項功能解決了上述提到的用戶違規性問題。

- 積分制度：

積分制度的設計可以在 Mobike 或是 oBike 看到，透過積分制度可以有效規範用戶，避免做出違反規定等不良行為，或是給予誘因，鼓勵用戶之間彼此互相監督。當用戶在結束騎乘時，未依規定將車輛停放在合法停車處、違反交通法則，或是忘記將車輛上鎖，甚至是添加個人私鎖時，都可以將用戶的積分扣除，暫停提供該用戶往後租借車輛的服務與權利；而相反的，如果用戶發現這些違規事項，協助回報車輛情況或停放地點等，則可以賺取積分，累積紅利換

取免費租車次數等其他好處。這項功能也替上述提到的用戶違規性問題提供更加完善的解決辦法。

- 標示車輛使用程度：

由於車輛來源為使用者自行提供，所以會產生單車種類款式眾多，車輛品質不一致的情況，但透過智能合約單車租借方式，用戶將可以透過合約自行將所提供的單車訂定押金與租金之金額，並在區塊鏈上的欄位記載車輛的使用程度、租借次數等相關資料，讓租借方能藉此當作得知車輛新舊程度的依據。此方式也能讓單車市場的租借雙方各取所需，用戶提供的車輛如果較為老舊，則可以透過訂定較低廉的租價，增加單車被租借的機會，而不在意車輛外觀新舊程度的用戶，則可能選擇此類型的單車進行租借；另一方面，如果用戶提供較為高檔的自行車時，也能因此要求較高額的租金進行租借服務，對於重視有舒適騎乘享受的使用者而言是一種保障。這種完全透明公開的方式，解決了共享單車之車輛品質的資訊不對稱問題，讓用戶不會因為租借到不符合自己需求喜好的車輛而影響騎乘體驗。

- 制訂騎乘範圍：

透過智能合約的撰寫，單車擁有人可以自行制訂該輛單車可供騎乘的距離，利用這種方式進而限制車輛的租借範圍。這種方式除了可以避免單車被租借到太過偏遠的地方，使擁有人不會憂慮車輛難以找回以外，也解決了車輛調度的問題，讓單車只會在規定的範圍內提供租借服務，確保一定範圍內的單車數量，以及降低運送單車所產生的成本。圖 3-2 說明智能合約單車租借之功能分別對應解決的問題。

問題	解決方法	說明
用戶違規性問題	押金制度、積分制度	透過押金制度提供單車擁有人之基本保障，並利用積分制度防止使用者的不當行為
租車問題	制訂騎乘範圍	讓單車只能在一定範圍內騎乘，確保車輛有一定的密集度
車輛調度問題	制訂騎乘範圍	讓單車只能在一定範圍內騎乘，而不會被租借到太過偏遠處，造成後續調度問題
車輛品質問題	標示車輛使用程度	使用者可依自身單車新舊程度，訂定不同租借價格，並選擇欲租借的車輛樣式

圖 3- 2：智能合約單車租借的功能說明

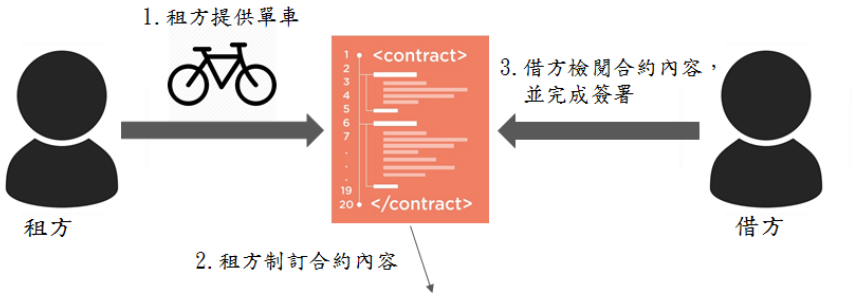
3.3 智能合約單車租借之服務流程

結合上一小節所提到的特色與功能，智能合約單車租借共有三項主要服務面向，分別是車輛註冊、車輛租借以及車輛歸還。透過車輛註冊服務，使用者可將自己的單車資產數位化，並提供租借服務予他人；而欲租借單車的使用者則透過簽訂智能合約，完成車輛租借與車輛歸還的服務，實現單車共享經濟之願景。以下將針對此三類服務進行介紹與流程說明。

3.3.1 車輛註冊

車輛註冊的服務，是讓有意提供單車給予他人進行租借的使用者，透過此功能將自己的車輛資訊註冊並紀錄在以太坊智能合約上。在這部份的合約內容當中，可以記載像是車輛編號、車輛廠牌、車輛持有人、車輛使用年份等資訊，讓相關的單車資料公開在合約上，讓欲租車的其他使用者能得知這些資訊。除此之外，進行車輛註冊的用戶，也要在這裡明訂押金的收取費用、租金的計價方式、是否有騎乘範圍的限制、或是其他額外的條件與規範，也就是說，所有有關租借車輛的相關資訊以及合約內容，都必須要在車輛註冊服務中，完整制訂此份智能合約。

對於進行車輛註冊服務、提供單車的用戶而言，合約內容可以是單方面立定的。儘管如此，當合約撰寫完成，並成功發佈在系統當中後，欲租借車輛的用戶可在上面看到完整公開的合約內容，再決定是否簽署此份合約並進行租借。簡而言之，租方與借方，在車輛註冊之階段時，所需要完成與注意的事情，分別是「制定合約內容」，以及「詳細檢閱合約內容並決定是否簽署」。圖 3-3 說明車輛註冊的服務流程。



欄位名稱	欄位資訊
車輛編號	7104029051
車輛持有人	廖子淳
車輛使用年份	2年
押金金額	10 ETH
租金計價方式	每半小時0.1 ETH
騎乘範圍	台北市中正區
…	…

圖 3- 3：車輛註冊的服務流程

3.3.2　車輛租借

在租借雙方確認合約內容無誤，並達成協議完成合約簽署之後，即可開始進行車輛租借的服務。在這個步驟當中，借方用戶首先必須與租方用戶確認車輛狀態，檢查是否有損壞的地方，當雙方都透過系統頁面確認無誤以後，接著再支付一筆先前租方用戶於合約上所規定的押金，此時智能合約會檢視自身的合約狀態，當區塊鏈上的節點通過認證押金已確實支付，達成共識之後，便會將車輛的使用權轉移到借方用戶，並將車輛解鎖，即可開始騎乘單車。

押金的部份，則會暫時存放在智能合約中，由智能合約代為保管，而非轉移到租方用戶的帳號底下。而在租借的騎乘過程中，除非借方用戶違反合約上所制訂的條件，否則租方用戶將無法干預騎乘者的車輛使用權，這項設計也可以替騎乘者帶來安全保障，避免發生騎乘狀態下卻被鎖車的意外。圖 3-4 說明車輛租借的服務流程。

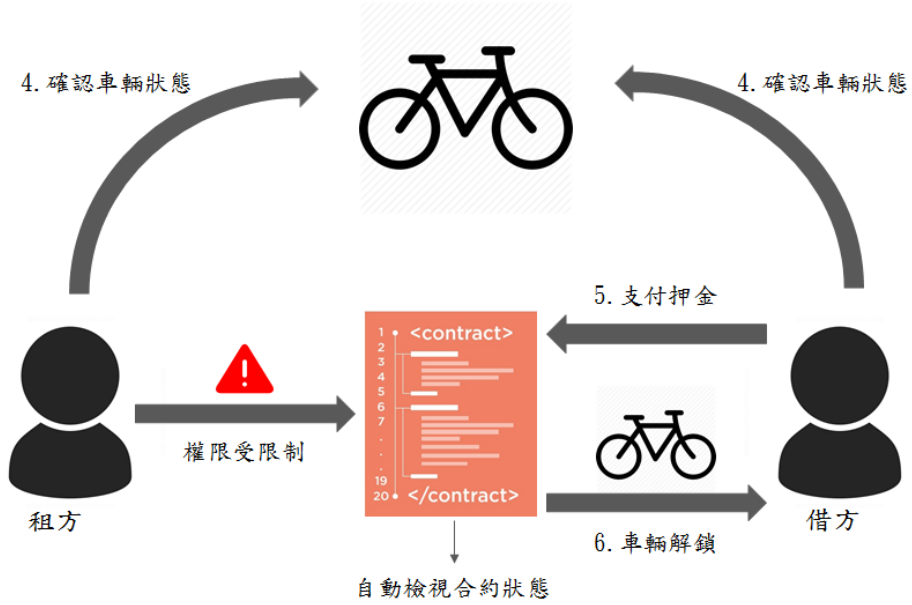


圖 3- 4：車輛租借的服務流程

3.3.3 車輛歸還

在車輛歸還的階段中，當借方用戶結束騎乘，此時智能合約會再度檢視自身的合約狀態，在確認車輛完成歸還後，智能合約會將車輛上鎖，並把車輛的使用權移轉回到借方用戶，同時也開始計算此次租借的費用，將租金移轉至租方用戶的帳號。而先前租車時所繳納的押金，會在下一位租車人租借時，確認車輛狀態沒有問題以後，由智能合約執行自動退還機制，把押金還給借方用戶，完成此次合約的履行與租借之服務。圖 3-5 說明車輛歸還的服務流程。

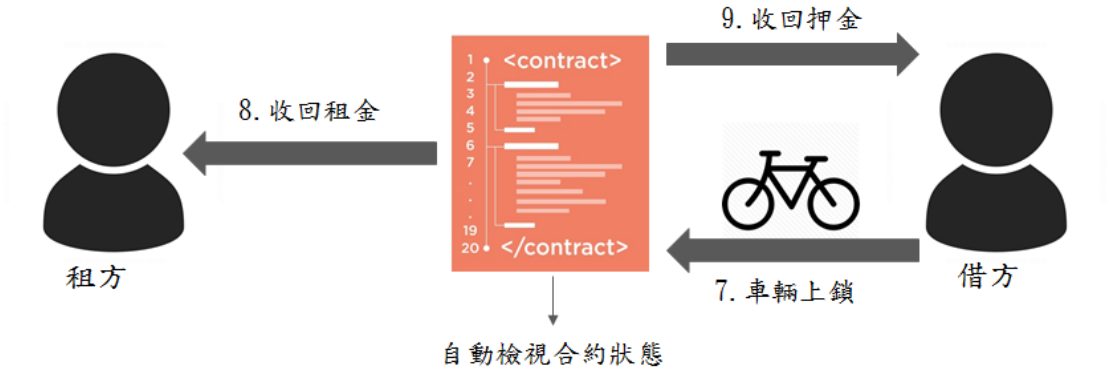


圖 3- 5：車輛歸還的服務流程

如果當下一位租車人 B 在確認車輛狀態時，發現車輛遺失或是有損毀的情況時，租方用戶將因為下一位租車人 B 的通報得知車輛發生狀況，此時智能合約將通知上一位借方用戶 A，把該借方用戶 A 的押金移轉到租方用戶的帳戶進行賠償。圖 3-6 說明押金補償的服務流程。

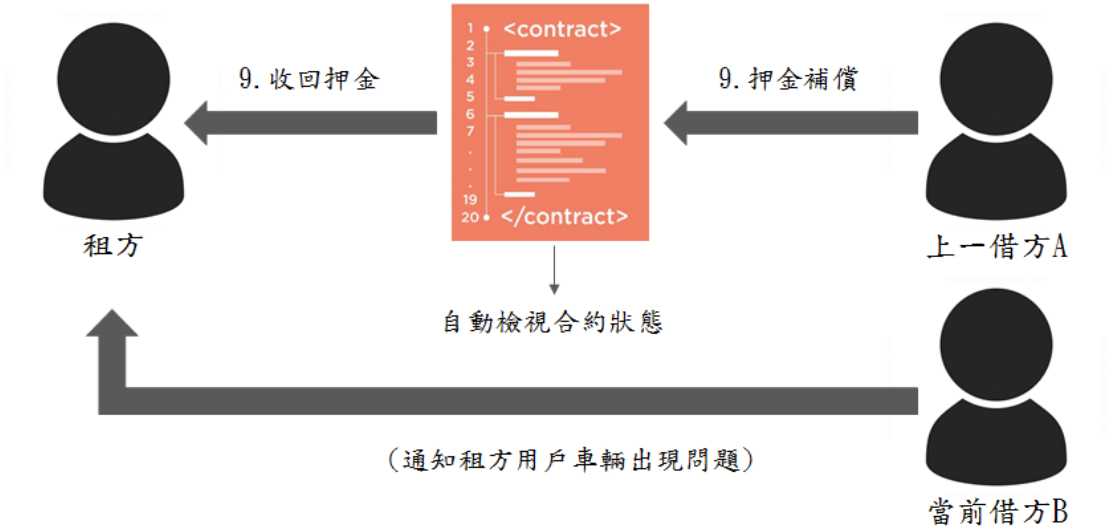


圖 3- 6：押金補償的服務流程

第四章 系統建置與評估

4.1 以太坊錢包

本研究將透過建置以太坊平台，下載由以太坊官方所發行的錢包軟體「Ethereum Wallet」所提供的功能與服務，撰寫一份單車租借合約，並實作智能合約單車租借之概念。Ethereum Wallet— 以太坊錢包，是一個連結以太坊區塊鏈上之去中心化應用程式的平台，它能讓使用者持有並確保以太幣，或是其他在以太坊上的加密資產，並且能夠撰寫、佈署以及使用智能合約。

4.1.1 安裝以太坊錢包

以太坊錢包軟體支援 Windows、Linux 與 Macosx 三種作業系統，而以下畫面步驟採用的是 64 位元的 Windows7 作業系統，安裝的以太坊錢包軟體版本是 0-8-10。

- 以太坊官方網址：https://www.ethereum.org/
- 以太坊錢包下載連結：https://github.com/ethereum/mist/releases
- 請根據電腦的作業系統選取相對應的軟體版本

1. 選取要安裝的版本功能：Test Network 或 Main Network；本次安裝選取的是 Test Network。

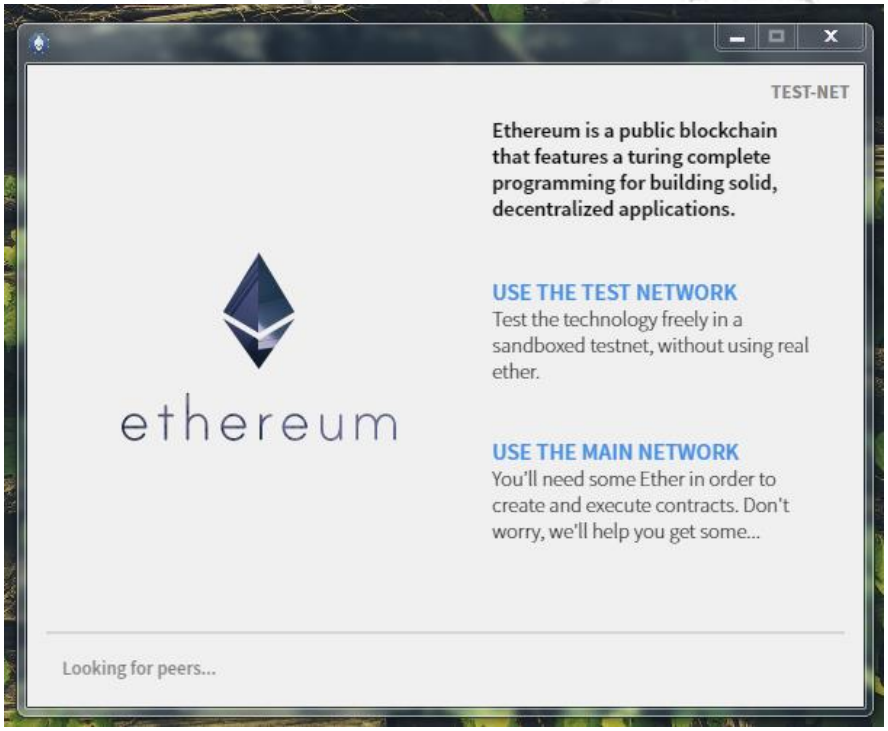


圖 4- 1：安裝以太坊錢包(1)

2. 輸入帳戶密碼：這邊的帳戶密碼十分重要，在安裝完成後進入主頁面時，許多功能的操作都需要輸入此組密碼作為驗證方式。如果忘記密碼，是沒有辦法透過電子信箱認證之類的方式重新輸入新密碼，可能會有遺失帳號的風險。

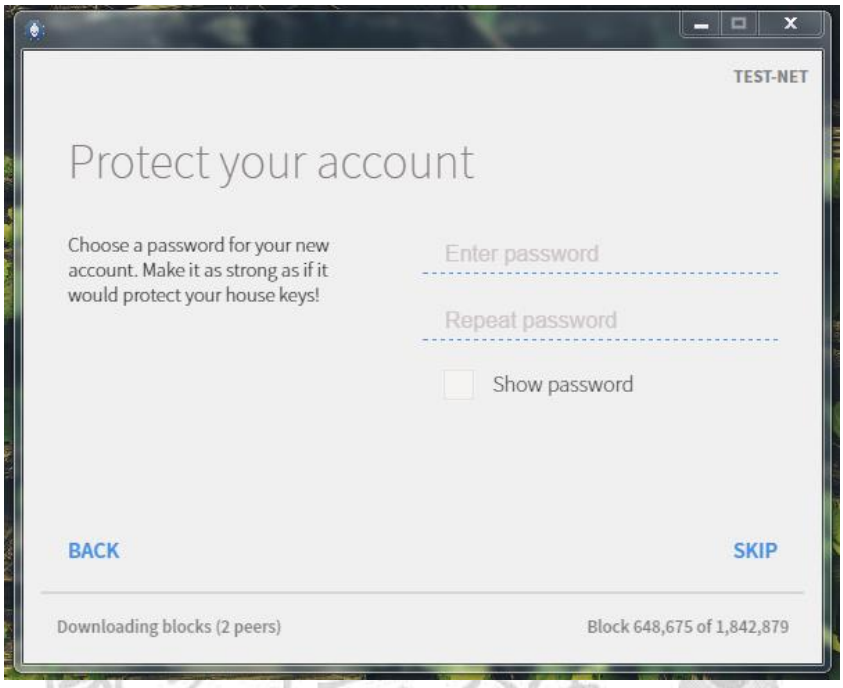


圖 4- 2：安裝以太坊錢包(2)

3. 等候下載完成：在這個部分需要同步區塊鏈資料，所以需要等待一段較長的安裝時間。

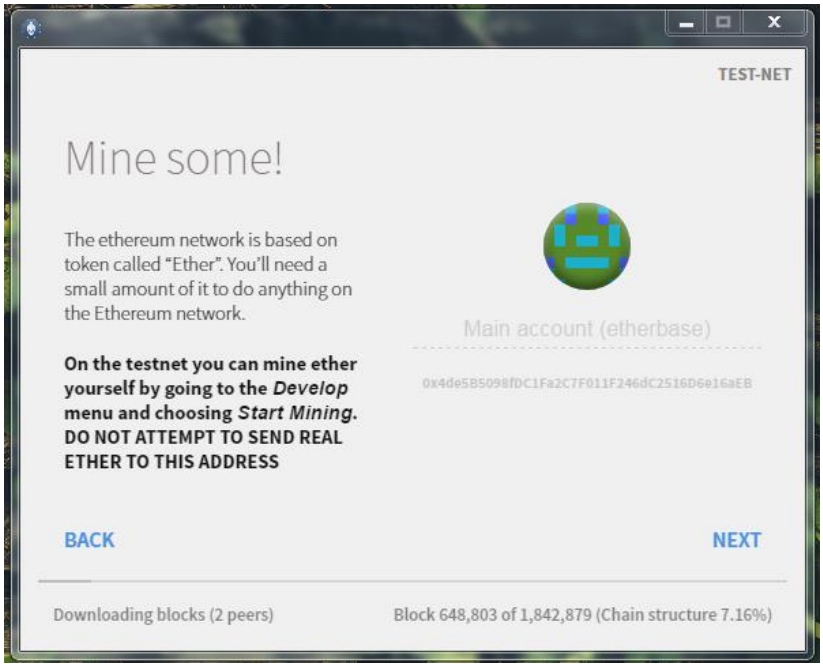


圖 4- 3：安裝以太坊錢包(3)

4.1.2　介面與功能

安裝完成後，進入到以太坊錢包的軟體主頁面，其中有三個主要功能項目，與一個顯示帳戶餘額的區域，位於主頁面上排，分別是 WALLETS、SEND、CONTRACTS 以及 BALANCE。圖 4-4 說明以太坊錢包的主頁面。

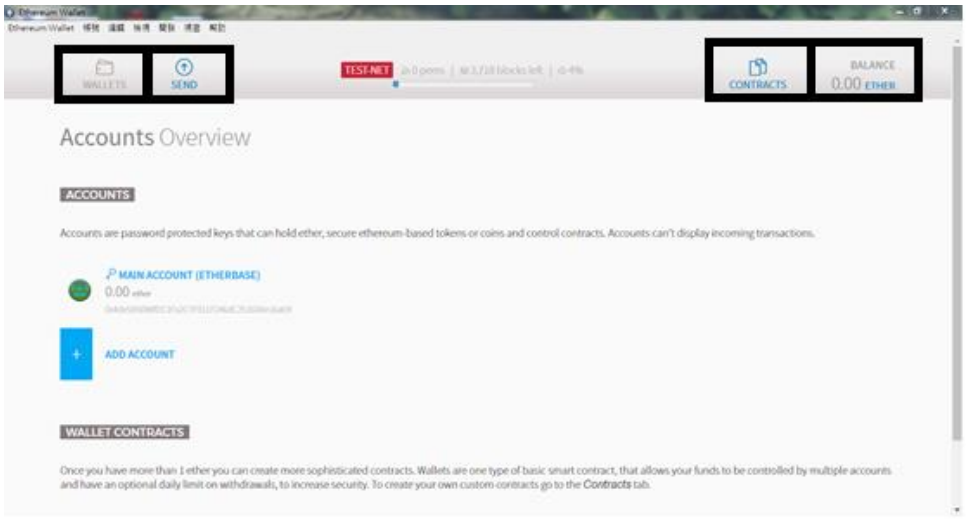


圖 4- 4：以太坊錢包的主頁面

- WALLETS：

  WALLETS 頁面同時也是以太坊錢包預設的首頁，是可以讓使用者查看相關帳戶資訊與交易紀錄的地方。在這個頁面當中，還有另外兩個功能，分別是 ADD ACCOUNT 以及 ADD WALLET CONTRACT。在 ADD ACCOUNT 中，使用者必須創建一個自己的帳戶，才能使用後續以太坊錢包所提供的服務，而帳戶則是可以保存以太幣，並保護其他以太坊基底的代幣(tokens)或貨幣，同時也能控制合約功能。圖 4-5 說明在創建完帳戶後，所看到的 MAIN ACCOUNT 頁面資訊，包含帳戶位址與帳戶餘額。

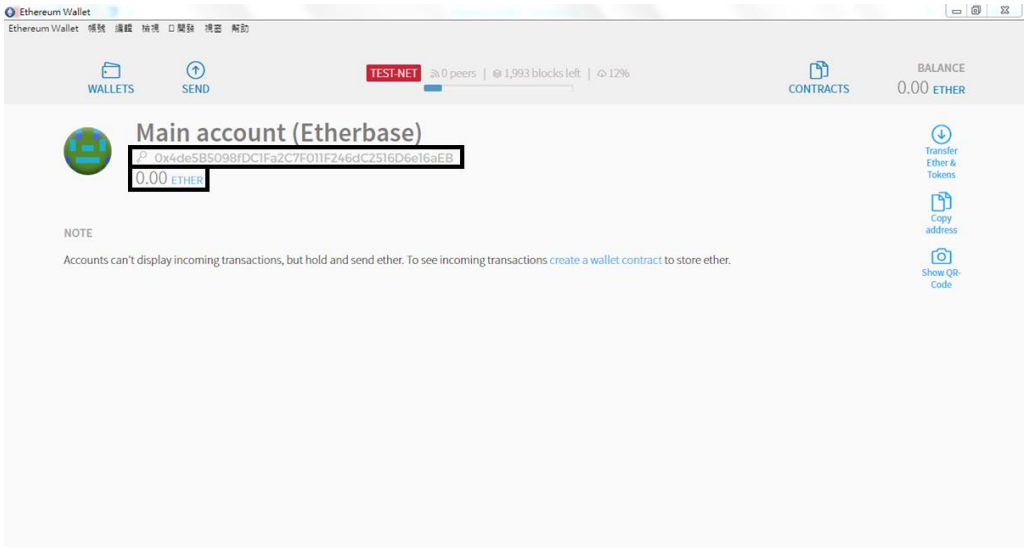


圖 4- 5：MAIN ACCOUNT 頁面

在 Accounts 中無法顯示正在進行中的交易，但可以執行接收、保存與發送以太幣的功能，如果要檢視正在進行中的交易，可以透過 create a wallet contract 功能儲存以太幣。而在 NEW WALLET CONTRACT 頁面中，使用者可以設定錢包合約的名稱、選擇持有者帳戶，還有錢包合約的種類(單一帳戶持有、多帳戶共有，或是匯入既有錢包)。圖 4-6 說明 NEW WALLET CONTRACT 頁面。

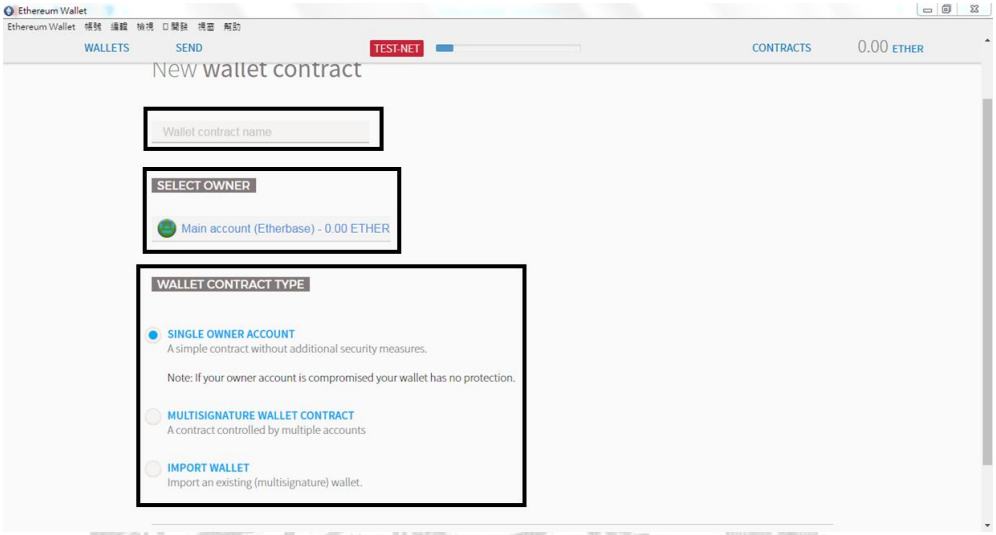


圖 4- 6：NEW WALLET CONTRACT 頁面

- SEND：

在 SEND 頁面中，使用者可以透過輸入帳戶位址代碼將錢幣發送給對方，並可以設定服務費率(FEE)調整交易速度，當使用者提供的服務費率越高時，完成交易的確認速度也會越快。圖 4-7 與 4-8 說明 SEND 頁面。

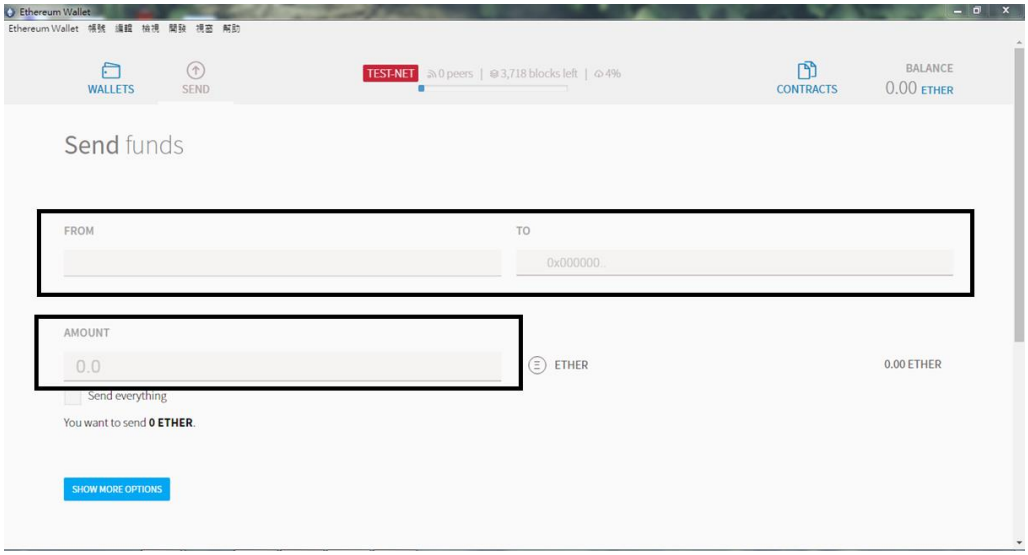


圖 4- 7：SEND 頁面- 輸入發送位址、目的位址、金額

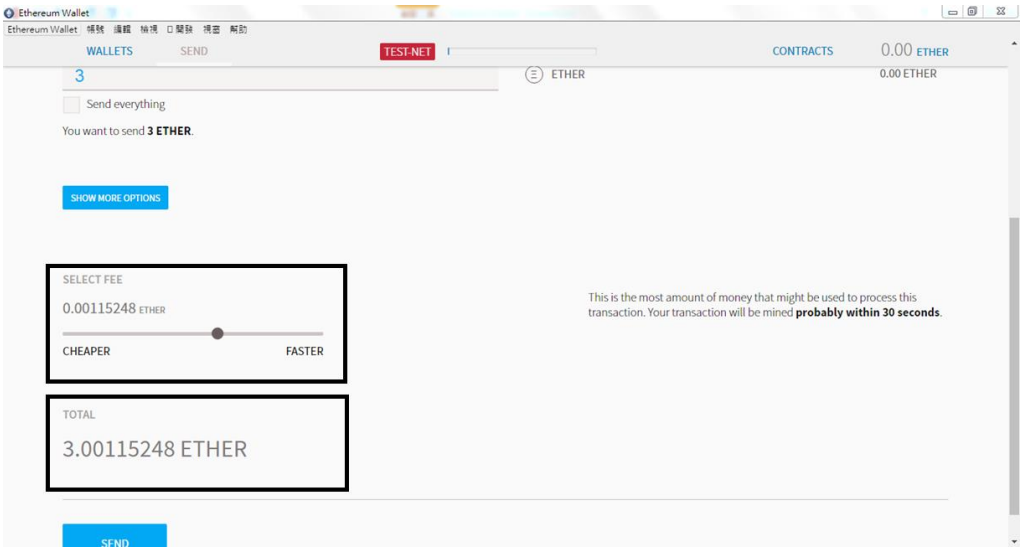


圖 4- 8：SEND 頁面- 選擇服務費率、顯示總金額

- CONTRACTS：

CONTRACTS 功能提供使用者客製化合約的服務，在這個頁面中有三個主要功能，分別是 DEPLOY NEW CONTRACT、WATCH CONTRACT 以及 WATCH TOKEN。圖 4-9 說明 CONTRACTS 頁面。

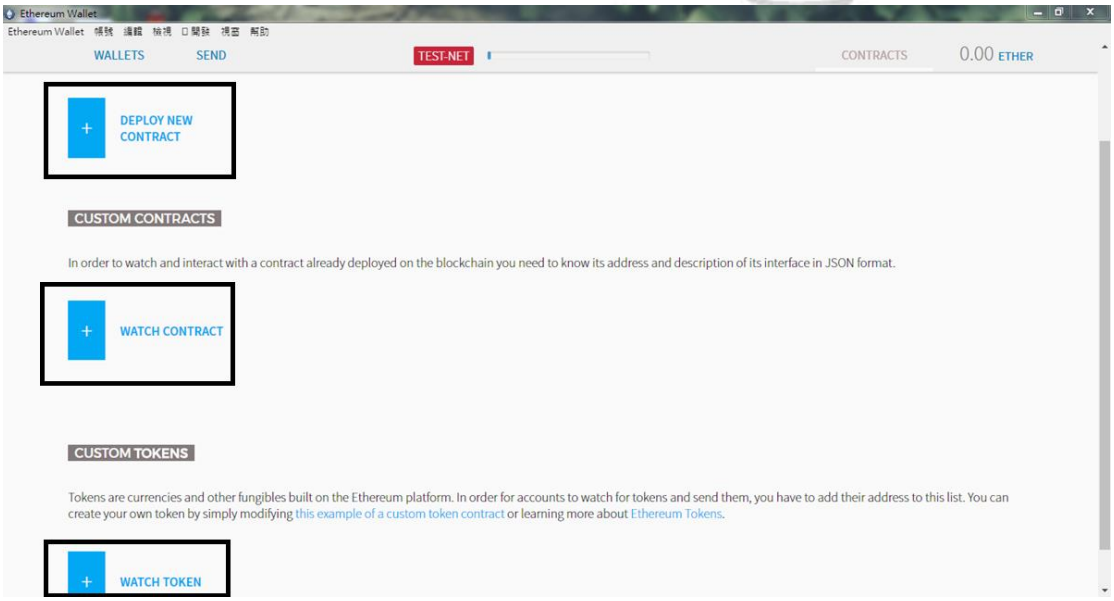


圖 4- 9：CONTRACTS 頁面

在 DEPLOY NEW CONTRACT 頁面中，使用者可以設定合約的發送位址與金額、撰寫合約程式碼、選擇服務費率，以及檢視部署合約的總成本。圖 4-10 與 4-11 說明 DEPLOY NEW CONTRACT 頁面。

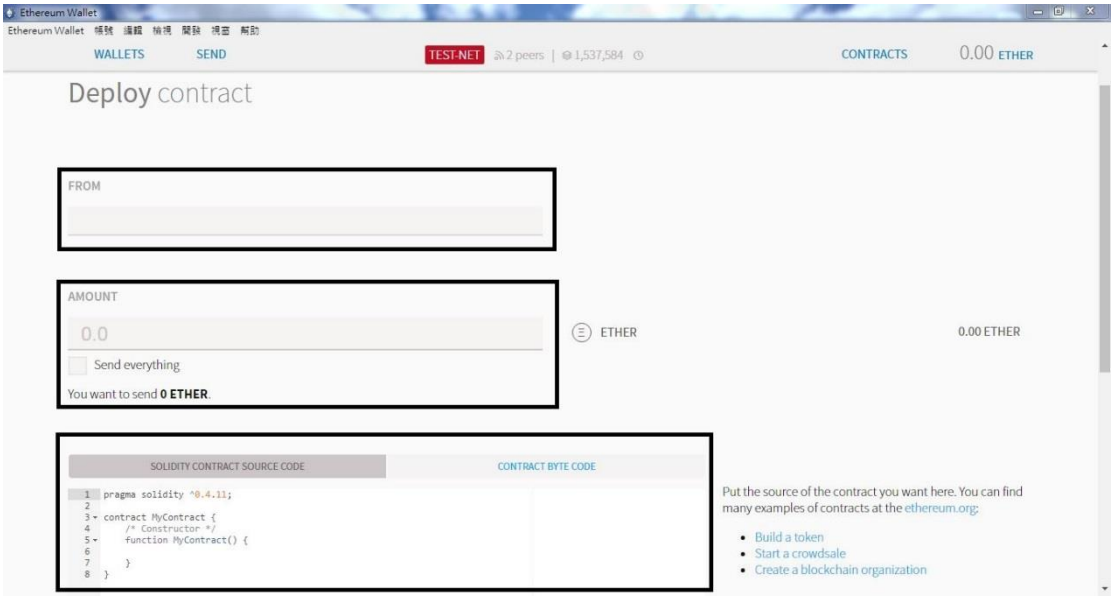


圖 4- 10：DEPLOY NEW CONTRACT 頁面- 選擇發送位址、金額、撰寫程式碼

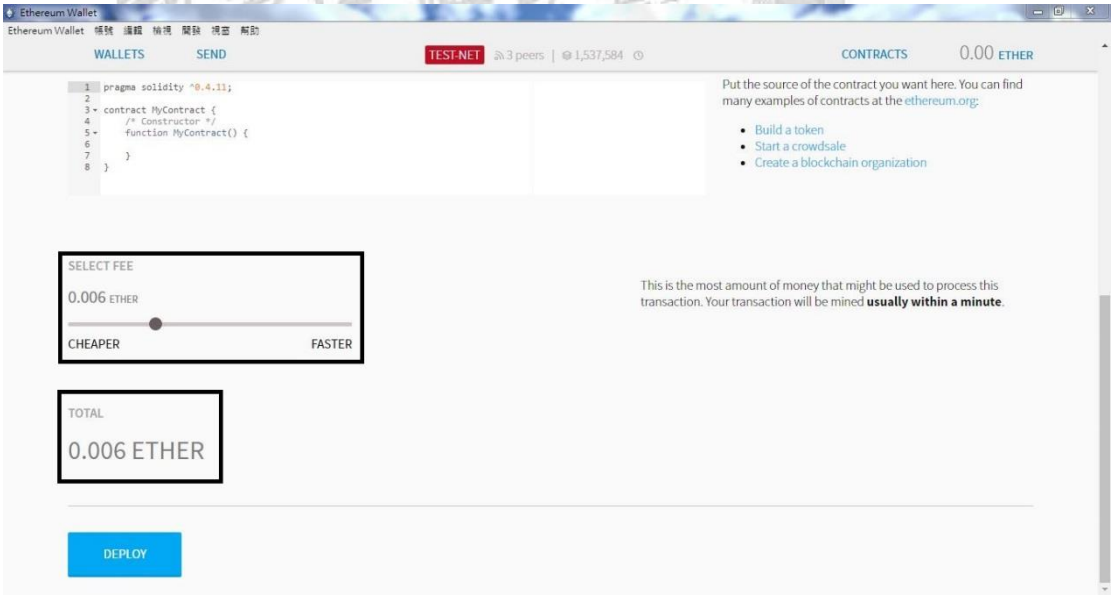


圖 4- 11：DEPLOY NEW CONTRACT 頁面- 選擇服務費率、顯示合約部署總成本

在 WATCH CONTRACT 頁面中，使用者可以透過輸入合約名稱、合約位址與 JSON 的介面格式來檢視合約資訊，也就是說，使用者必須得知該合約的位址以及用 JSON 格式表式的介面敘述，才能檢視已發佈的合約。圖 4-12 說明 WATCH CONTRACT 頁面。

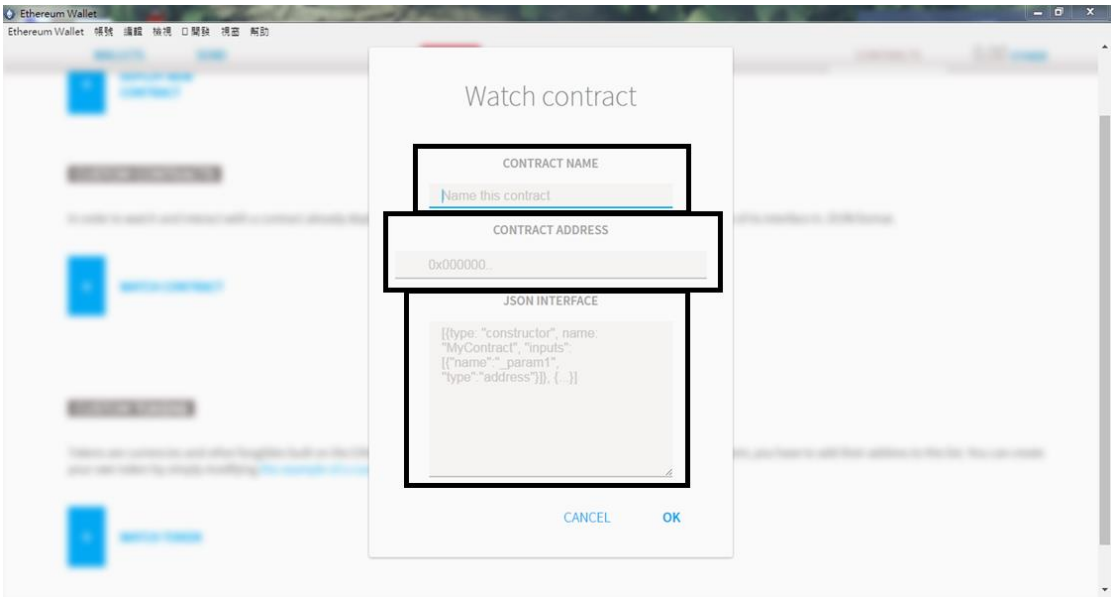


圖 4- 12：WATCH CONTRACT 頁面

在 WATCH TOKEN 頁面中，使用者可以藉由輸入代幣合約位址、代幣名稱、代幣符號以及代幣的最小切割單位，於以太坊平台自行創建新的數位代幣。為了要使帳戶能檢視並發送這些 Tokens，使用者必須將它們的帳戶位址加入此代幣合約中，並透過修改下列的客製化代幣合約範例，創建屬於自己的 token。圖 4-13 說明 WATCH TOKEN 頁面。

圖 4- 13 ：WATCH TOKEN 頁面與 token 設計預覽

4.2 撰寫合約

以下將說明智能合約單車租借的相關功能以及合約內容，主要架構分為三部分，分別是：車輛註冊的功能說明、合約的初始化格式說明，以及合約的主要功能說明。其中，車輛註冊的車輛資訊功能說明，會透過備註的型式撰寫在智能合約當中，希望在日後利用開發 APP 的方式提供服務，將車輛資訊介面化呈現。而合約的初始化格式說明，以及合約的主要功能說明之部分，則會透過以太坊錢包的 CONTRACTS 頁面撰寫呈現，並將針對合約的語法格式以及程式碼的功能進行解說。圖 4-14 說明智能合約單車租借的系統架構圖。

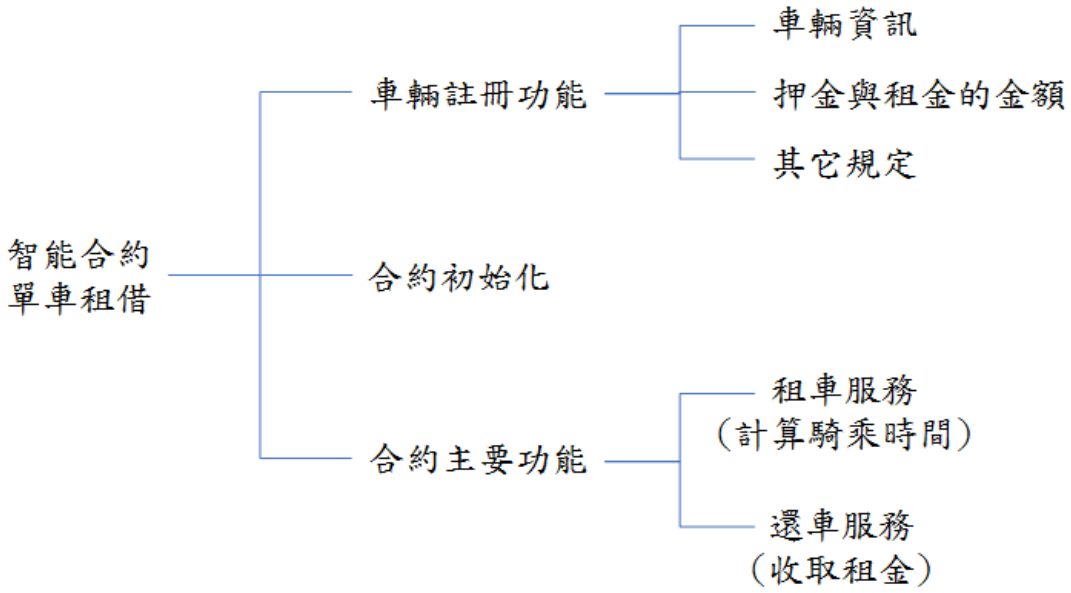


圖 4- 14：智能合約單車租借的系統架構圖

4.2.1 車輛註冊功能

車輛註冊功能的設計目的，是為了讓用戶能將自己的單車進行註冊，並提供租借服務予其他用戶，同時也讓想租借單車的用戶能對即將騎乘的車輛資訊一目了然。在這裡顯示的車輛資訊，包含：車輛編號、車輛持有人、車輛使用年份等，以及押金金額、租金計價方式，還有騎乘範圍等其它規定。車輛註冊功能在未來可以透過 APP 的開發，設計一套易於用戶操作的使用者介面，並將註冊完的車輛資訊存放在區塊鏈上，讓單車持有人與單車租借人都能方便查看。圖 4-15 說明車輛註冊的車輛資訊備註，與押金租金的金額制定等其它規定。

```
//bikeID = "7104029051"
//bikeowner = "廖子淳"
//bikedepreciation = "2 years"
//deposit = "10 eth"
//rent = "0.1 eth/0.5 hr"
//ridingrange = "台北市中正區"

pragma solidity ^0.4.11;

contract bikeRenting {
    address public owner;
    address public currentRenter;
    uint public expireTime;
    uint public unitPrice;
    function bikeRenting(uint _unitPrice){
        owner = msg.sender;
        currentRenter = 0x0;
        expireTime = now;
        unitPrice = _unitPrice;
    }
    function  rent() payable returns(bool){
        if(currentRenter != 0x0) return false;
        else{
            if(   (msg.value/1 ether)  / unitPrice   <   1) return false;
            else {
                expireTime = now + 30 minutes * (msg.value/1 ether)/unitPrice;
                currentRenter = msg.sender;

            }
        }
```

圖 4- 15：車輛資訊備註，包含車輛編號、車輛持有人、車輛使用年份等，以及押金金額、租金計價方式，還有騎乘範圍。

4.2.2 合約初始化

合約的開頭包含合約的狀態宣告，以及合約的初始化兩部分。在合約的狀態宣告部分，有一些 solidity 語言在智能合約上的專有變數，像是 ether、finney、wei 等交易單位，或是 seconds、weeks、years、now 等時間用法，而其中比較特別的是 now，代表的是當前區塊的時間。除此之外，還有像是 msg 與 address 的用法。

- msg：代表的是當前交易的相關資訊
- msg.sender：當前交易的發起人
- msg.value：當前交易所附帶的金額
- address：代表的是一個位址
- address.balance：查詢此 address 的餘額
- address.send(amount)：發送貨幣到此 address

而在撰寫合約前，首先必須先宣告所使用的 solidity 語言版本，接著才開始撰寫合約內容。bikeRenting 為該份合約的名稱，接著並開始進行狀態宣告，包括 owner(租方)、currentRenter(借方)、expireTime(租借到期時間)以及 unitPrice(一單位的錢) 。圖 4-16 說明合約的狀態宣告。

```
pragma solidity ^0.4.11;

contract bikeRenting {
    address public owner;
    address public currentRenter;
    uint public expireTime;
    uint public unitPrice;
    function bikeRenting(uint _unitPrice){
        owner = msg.sender;
        currentRenter = 0x0;
        expireTime = now;
        unitPrice = _unitPrice;
    }
    function  rent() payable returns(bool){
        if(currentRenter != 0x0) return false;
        else{
            if(   (msg.value/1 ether)  / unitPrice   <   1) return false;
            else {
                expireTime = now + 30 minutes * (msg.value/1 ether)/unitPrice;
                currentRenter = msg.sender;

            }
        }
    }
    function  isInUse(bool ifWantToRent) payable returns(bool){
        if( expireTime > now ) return true;
        else{
            currentRenter = 0x0;
            if( ifWantToRent == true) rent();
```

圖 4- 16：合約的狀態宣告

結束合約的狀態宣告後，接著便是進行合約的初始化功能撰寫，這部分包含設定先前所宣告的 owner、currentRenter、expireTime 以及 unitPrice 的值。其中，owner 的值為該合約的發起人，currentRenter 的值為 0x0(即空值)，expireTime 的值為 now。圖 4-17 說明合約的初始化。

```
pragma solidity ^0.4.11;

contract bikeRenting {
    address public owner;
    address public currentRenter;
    uint public expireTime;
    function bikeRenting(uint _unitPrice){
        owner = msg.sender;
        currentRenter = 0x0;
        expireTime = now;
        unitPrice = _unitPrice;
    }
    function  rent() payable returns(bool){
        if(currentRenter != 0x0) return false;
        else{
            if(  (msg.value/1 ether) / unitPrice < 1) return false;
            else {
                expireTime = now + 30 minutes * (msg.value/1 ether)/unitPrice;
                currentRenter = msg.sender;

            }
        }
    }
    function  isInUse(bool ifWantToRent) payable returns(bool){
        if( expireTime > now ) return true;
        else{
            currentRenter = 0x0;
            if( ifWantToRent == true) rent();
        }
```

圖 4- 17：合約的初始化

4.2.3 合約主要功能

合約的主要功能分為三部分，分別是 rent、inUse 以及 rentingFee。Rent 的功能為檢查租借條件，而 inUse 的功能為檢查租借狀態，rentingFee 的功能則是收取租借費用。

- function rent()

  在 rent() 功能當中，首先智能合約會檢查 currentRenter 的值是否為 0x0，也就是檢查該車輛是否正在被租借。當確定沒有被租借之後，接著智能合約會檢查 unitPrice 是否大於 1，即檢查租借者是否有給足一單位的錢。最後，智能合約會設定租借時間，將租借到期時間的值加上可租借的時間，並將 currentRenter 的值指定給該租借者，且記錄該租借者為誰、當前區塊的時間，以及所交易的金額。圖 4-18 說明 function rent()之步驟。

```
14      uint public unitPrice;
15      function bikeRenting(uint _unitPrice){
16          owner = msg.sender;
17          currentRenter = 0x0;
18          expireTime = now;
19          unitPrice = _unitPrice;
20      }
21      function rent() payable returns(bool){
22          if(currentRenter != 0x0) return false;
23          else{
24              if(  (msg.value/1 ether) / unitPrice < 1) return false;
25              else {
26                  expireTime = now + 30 minutes * (msg.value/1 ether)/unitPrice;
27                  currentRenter = msg.sender;
28              }
29          }
30      }
31      function inUse(bool ifWantToRent) payable returns(bool){
32          if( expireTime > now ) return true;
33          else{
34              currentRenter = 0x0;
35              if( ifWantToRent == true) rent();
36          }
37      }
38      function rentingFee(){
39          if( msg.sender == owner){
40          owner.transfer(this.balance) ;
41          }
42      }
43  }
```


圖 4- 18：function rent()

- function inUse()

  在 inUse() 功能當中，智能合約將再次檢查車輛狀態是否為租借中，如果 expireTime 的值大於 now 的值，則表示過期時間在未來，代表正在租借期間，否則便是可供租借的狀態，這時 currentRenter 的值為 0x0，並且智能合約將會檢查呼叫函式的人是否也要進行租借，若回傳的值為 true，則執行 rent() 功能。圖 4-19 說明 function inUse()之步驟。

```
14      uint public unitPrice;
15      function bikeRenting(uint _unitPrice){
16          owner = msg.sender;
17          currentRenter = 0x0;
18          expireTime = now;
19          unitPrice = _unitPrice;
20      }
21      function rent() payable returns(bool){
22          if(currentRenter != 0x0) return false;
23          else{
24              if(  (msg.value/1 ether) / unitPrice < 1) return false;
25              else {
26                  expireTime = now + 30 minutes * (msg.value/1 ether)/unitPrice;
27                  currentRenter = msg.sender;
28              }
29          }
30      }
31      function inUse(bool ifWantToRent) payable returns(bool){
32          if( expireTime > now ) return true;
33          else{
34              currentRenter = 0x0;
35              if( ifWantToRent == true) rent();
36          }
37      }
38      function rentingFee(){
39          if( msg.sender == owner){
40          owner.transfer(this.balance) ;
41          }
42      }
43  }
```

圖 4- 19：function inUse()

之所以需要 inUse() 功能來讓下一個用戶確認車輛的租借狀態，是因為智能合約並非真的完全「智能」，而是需要依靠外界觸發並呼叫函式才能執行合約，所以當執行 inUse() 功能的用戶，因付出成本檢查了車輛之狀態時，必須要確保該檢查的人如果也想租借車輛，則可以優先進行租借服務。

- function rentingFee()

  在 rentingFee() 功能當中，智能合約會檢查執行此功能交易的發起人是否為 owner，若結果為是則將租借人的租金發送至 owner 帳戶，確保只有 owner 可以存取此筆費用。圖 4-20 說明 function rentingFee()之步驟。

```
14      uint public unitPrice;
15      function bikeRenting(uint _unitPrice){
16          owner = msg.sender;
17          currentRenter = 0x0;
18          expireTime = now;
19          unitPrice = _unitPrice;
20      }
21      function rent() payable returns(bool){
22          if(currentRenter != 0x0) return false;
23          else{
24              if(  (msg.value/1 ether) / unitPrice < 1) return false;
25              else {
26                  expireTime = now + 30 minutes * (msg.value/1 ether)/unitPrice;
27                  currentRenter = msg.sender;
28              }
29          }
30      }
31      function inUse(bool ifWantToRent) payable returns(bool){
32          if( expireTime > now ) return true;
33          else{
34              currentRenter = 0x0;
35              if( ifWantToRent == true) rent();
36          }
37      }
38      function rentingFee(){
39          if( msg.sender == owner){
40          owner.transfer(this.balance) ;
41          }
42      }
```

圖 4- 20：function rentingFee()

4.3 部署合約

撰寫完合約程式碼之後，接著就是將這份合約發佈到區塊鏈網路中。首先必須先選擇要部署的合約，即 bike Renting，再輸入 unit price 的值，在這邊設定為 1。其中，unit price 就是單次租借所需的金額費用。圖 4-21 至 4-24 說明部署合約

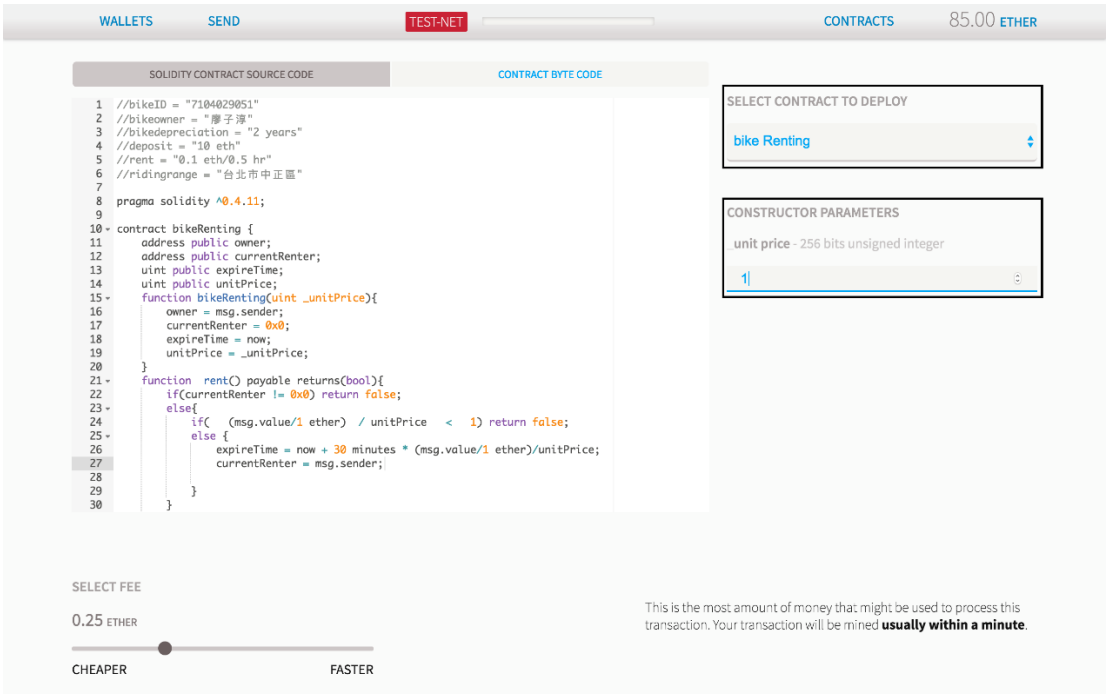


圖 4- 21：選擇欲部署的合約

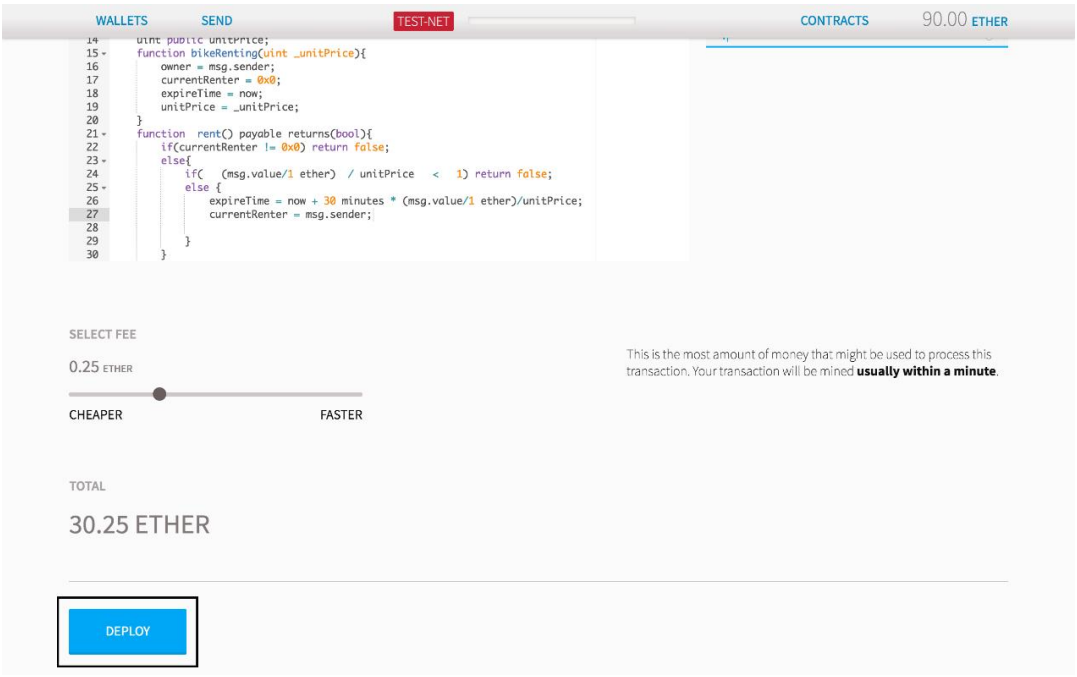


圖 4- 22：進行部署

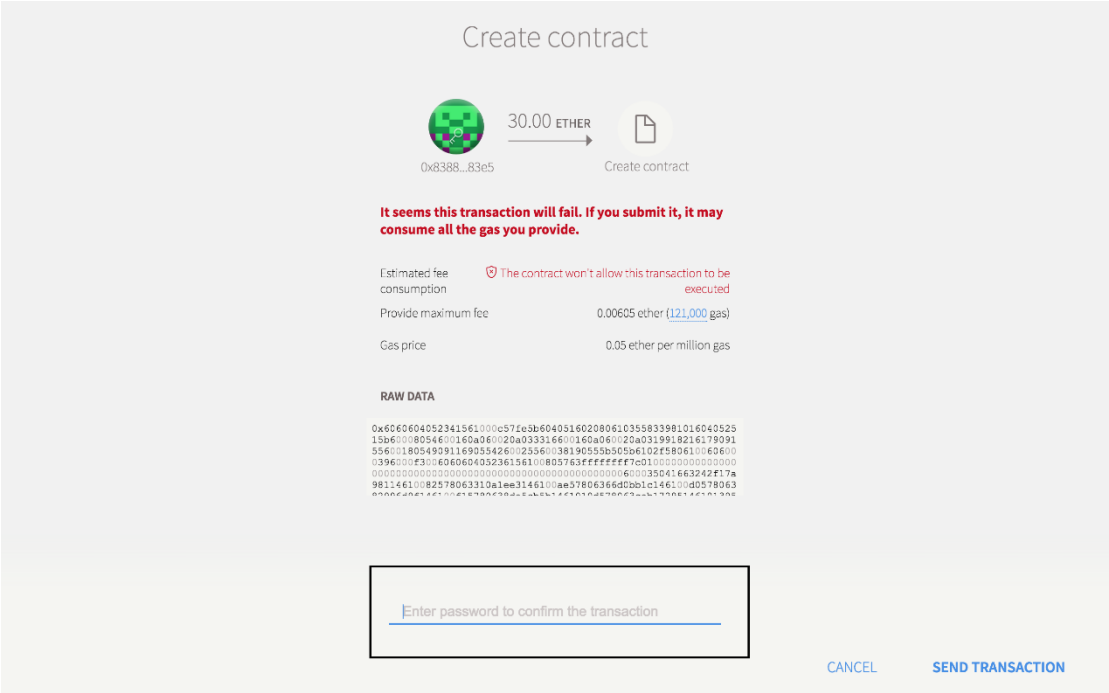


圖 4- 23：輸入密碼確認執行部署

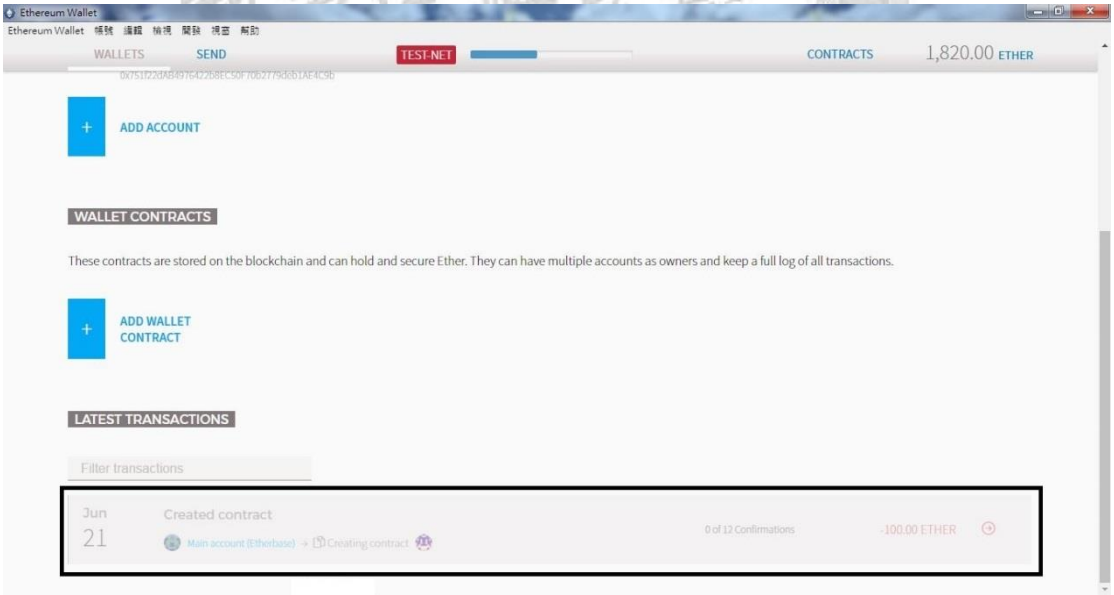


圖 4- 24：等待確認合約發布，當系統經過 12 個驗證節點確認後，即完成合約發佈。

4.4 執行合約

完成合約部署之後，接著可以在 CONTRACTS 頁面檢視剛剛佈署的合約。在畫面左方可以看到的是合約的狀態，顯示的資訊包括 Current renter、Expire time、Owner 以及 Unit price。而在畫面右邊則可以選取欲執行的功能，其中包括 Rent、In Use 以及 Renting Fee。圖 4-25 為 bike Renting 合約的頁面資訊。

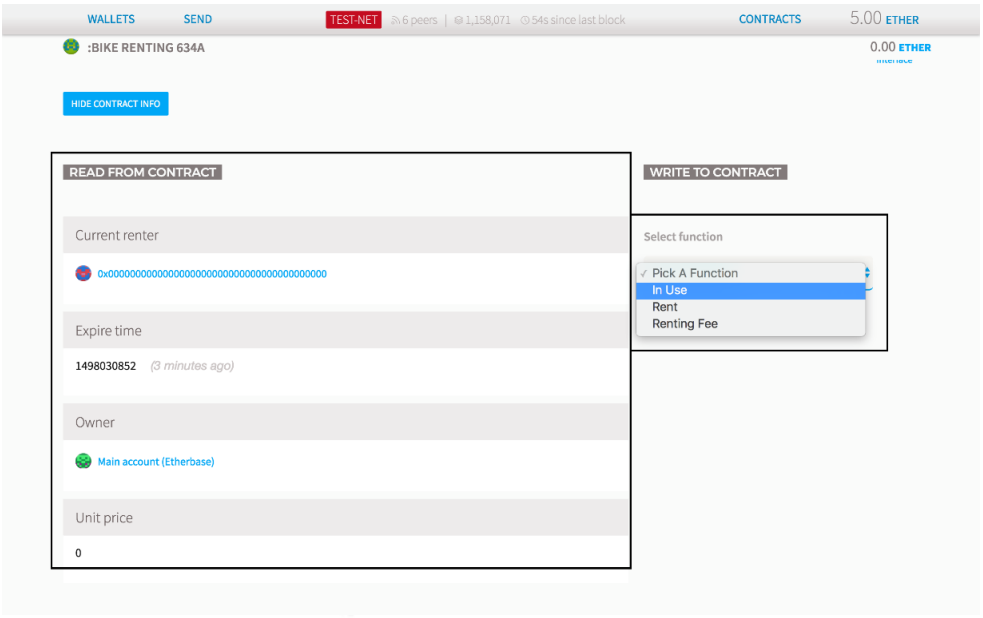


圖 4- 25：BIKE RENTING 頁面資訊

當執行 Rent 功能時，其中的 Execute from 欄位代表的是借方用戶的錢包位址，由於租借人不一定只擁有一個錢包帳戶，所以在這邊可以選擇欲進行扣款的帳戶為何；而 Send 欄位代表的則是發送的金額，也就是欲租借的時間，金額越高即租借時間越久。圖 2-26 說明 Rent 功能之執行。

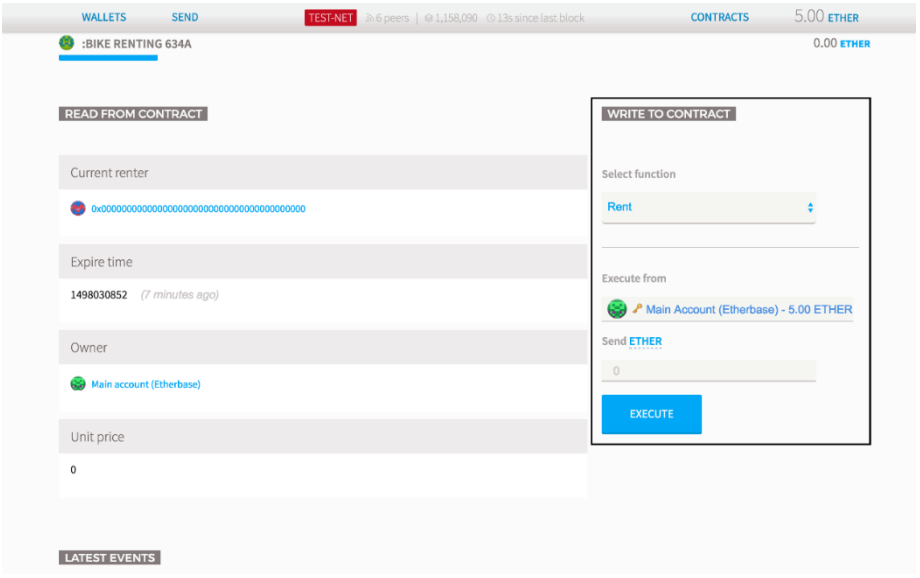


圖 4- 26：執行 Rent 功能

當執行 In Use 功能時，首先可以先點選是否也要進行租借，接著並透過 Execute from 欄位選擇欲進行租借的帳戶錢包，最後再輸入欲租借的金額。In Use 功能的目的是讓使用者得知目前單車的使用情況，幫助觸發智能合約驗證合約狀態，並且讓執行此功能的用戶擁有優先租借權。圖 2-27 說明 In Use 功能之執行。

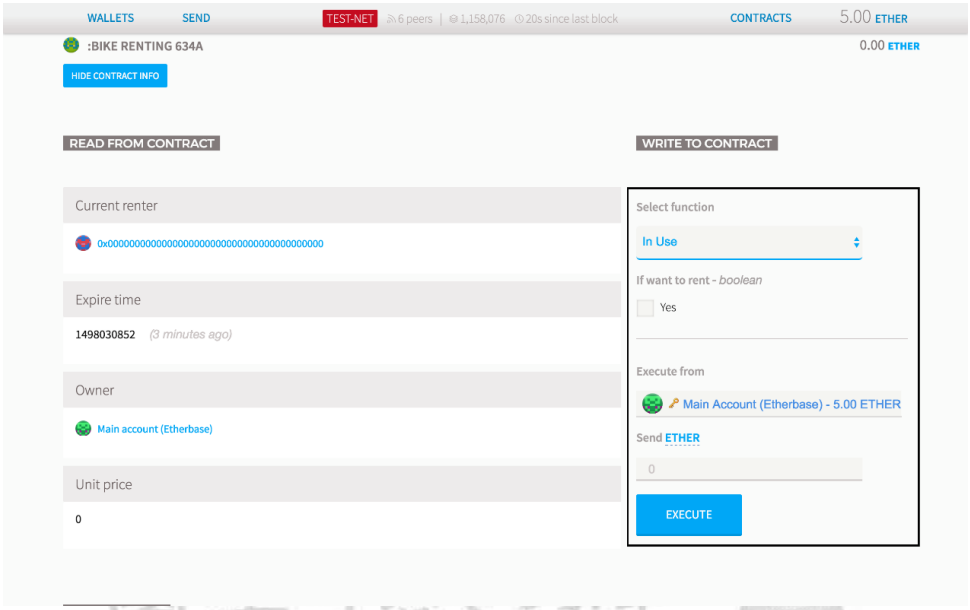


圖 4- 27：執行 In Use 功能

當執行 Renting Fee 功能時，Execute from 欄位代表的是欲存入金額的帳戶錢包為何，而 Send 欄位代表的則是欲存入的金額。圖 2-28 說明 Renting Fee 功能之執行。

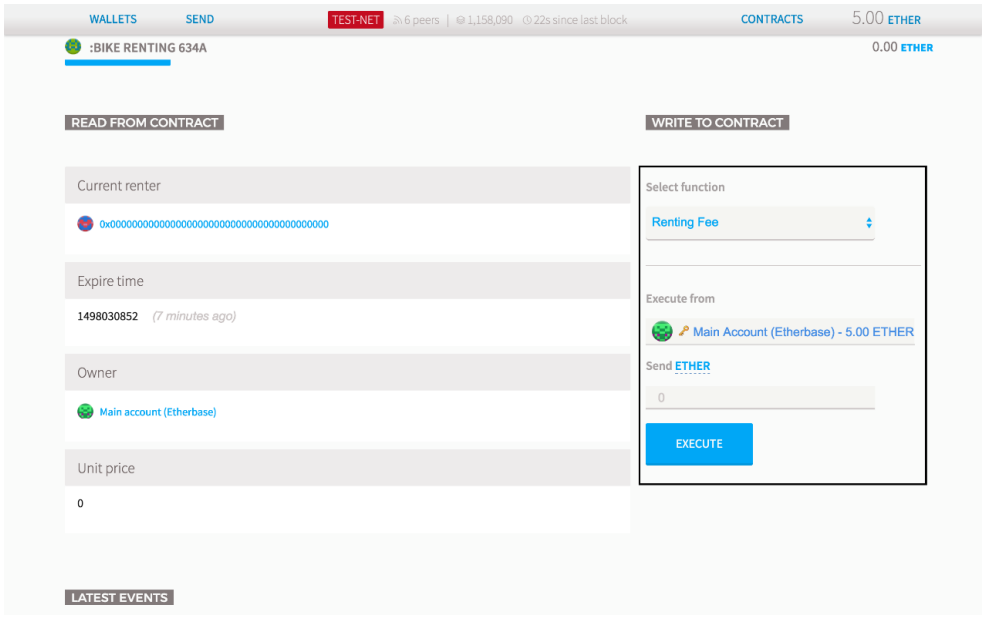


圖 4- 28：執行 Renting Fee 功能

最後，在選擇欲執行的功能之後，按下功能欄位下方的 EXECUTE 鍵，即完成了合約的執行步驟。在這個步驟當中，一樣也是需要輸入用戶密碼確認執行，並等待驗證節點進行確認。圖 2-29 以執行 Rent 功能為例，說明合約的最終執行步驟。

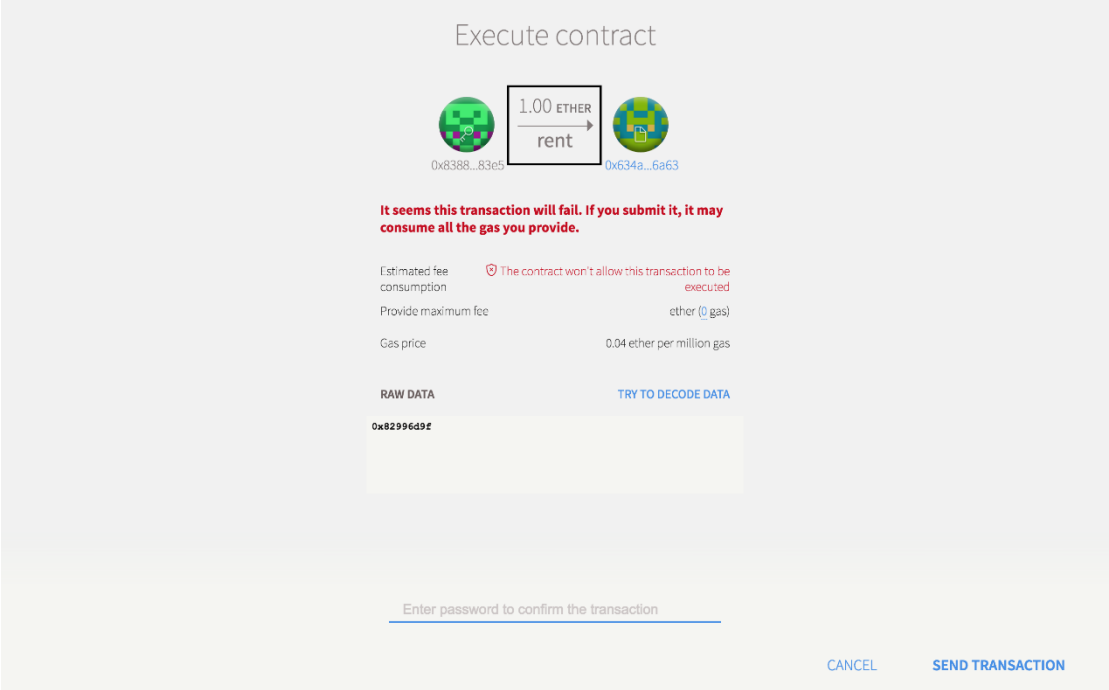


圖 4- 29：執行合約的最後一步

第五章 結論與建議

5.1 智能合約單車租借的優勢

智能合約單車租借，結合無站點式的單車租借模式，以及透過智能合約打造創新的租借流程，為實現真正的單車共享經濟提供了一套方式與機會。除了解決傳統固定站點式之停車車位不足的問題以外，同時也針對目前逐漸普及的無站點式之車輛停放的問題等，提出利用智能合約的解決辦法。智能合約單車租借解決的問題包括：

- 固定站點式在還車時可能會發生的車位不足問題。
- 無站點式在車輛停放時所引發的違停問題或其他用戶違規問題。
- 無站點式在租車時不方便找尋的問題。
- 無站點式在車輛調度的高成本問題。
- 透過用戶提供車輛所可能引發的車輛品質問題。

而以上之問題提出，利用智能合約單車租借的解決辦法分別為：

- 採取無站點式的單車租借模式。
- 押金制度、積分制度：透過押金制度提供單車擁有人之基本保障，並利用積分制度防止使用者的不當行為。
- 制訂騎乘範圍：讓單車只能在一定範圍內騎乘，確保車輛有一定的密集度。
- 制訂騎乘範圍：讓單車只能在一定範圍內騎乘，而不會被租借到太過偏遠處，造成後續調度問題。
- 標示車輛使用程度：使用者可依自身單車新舊程度，訂定不同租借價格，並選擇欲租借的車輛樣式。

而利用簽訂智能合約，讓用戶可以將本身閒置的自行車，提供給其它有需求的使用者進行單車租借的服務，這種方式比起傳統的固定站點式，或是無站點式之營運模式，都來得更符合共享經濟的概念，因為這種方式不再是只侷限於透過中央營運商或大型企業來提供單車租借服務，而是每個參與服務的用戶都視一個營運商，讓一般大眾都有機會成為實質的獲利者。

另一方面，這個模式的優勢在於營運者本身不需要考慮太多單車成本和運營管理的問題，只需要透過建構技術平台媒合各個使用者，讓欲採用智能合約進行單車租借服務的現有企業營運商，可以大幅降低各種營運上的硬體成本以及管理方面的維護問題。

5.2 智能合約單車租借的劣勢

儘管智能合約單車租借的服務概念，看似替現有之各類單車租借市場上所遇到的不同問題提供解決辦法，但要真正實現透過智能合約的方式，完成單車租借服務，實際上仍然面臨一些需要被克服的考驗。主要的問題像是：

1. 合約制訂方面的相關問題
2. 車輛硬體裝置的成本問題
3. 車輛損壞的認定問題
4. 區塊鏈交易確認的租借速度問題

這些問題都是影響智能合約單車租借服務，是否具真正有效性與實用性的因素，以下將針對這四類問題進行討論。

- 合約制訂的多樣化

原則上來說，如果透過智能合約的方式進行單車租借服務，那麼每一輛車都有一份需要簽署的對應租借合約，也就是說，每當用戶要進行租借時，都必須先檢視過合約內容以及規範，若確認沒問題且符合自己的使用需求後，才會進一步簽署合約並租車騎乘。這種「有條件」的租借方式將迫使使用者需額外費心留意合約規範，且每租一次車就必須重新檢視合約內容，因為每個車輛擁有人所制訂的合約都可能不一樣，而這比起現行的單車租借模式，可說是來的更加複雜繁瑣，因此勢必會對使用者的租車意願造成衝擊。

- 車輛硬體裝置的成本

要能實現智能合約單車租借服務，除了外在環境需要有網路訊號之外，具備能夠辨識資訊、處理指令的智能鎖，以及提供定位服務的 GPS 裝置，更是不可或缺的重要因素。但現有市面上的自行車款式不太會有這樣的規格配備，更別說是一般大眾所使用的單車要能符合這樣的需求，所以，提供智能合約單車租借服務的車輛必須額外添加智能鎖與 GPS 裝置，但這對還未獲利就先額外付出花費的車輛用有人來說，除了可能降低提供車輛進行租借服務的意願之外，也無疑是一項成本負擔。

- 車輛損壞的認定

當租借時發生車輛損壞的情形，要如何歸咎責任，甚至是要求賠償，這部分的認定與處理將會是一個難解的問題。舉例來說，如果在車輛租借前沒有發現單車有異狀，而是在正常騎乘下才突然發生問題，這時候到底是要判定為使用者有不當的操作行為才導致損壞，還是車輛零件老舊本身就該汰換檢修? 而在另一種情況下，當前一位租借人 A 完成還車後，下一位欲租車人 B 卻發現車輛有損壞的情況，此時除了可能是上一位租借人 A 造成的損壞之外，也有可能是在 A 還車後與 B 租車前的這段期間，遭到其他人破壞，在這種情況下，智能合約將很難真正實現公正第三方的角色，如果貿然把 A 的押金沒收顯然是不洽當的做法。

- 租借確認的執行速度

合約的簽定與確認，均是透過區塊鏈技術執行，也就是說，區塊的認證速度會直接影響租借流程的速度。如果智能合約單車租借無法做到「簽約完即可騎走」的交易速度，那麼將不會有人願意使用這項服務，畢竟不會有使用者可以接受租借單車還要先耗費時間等待交易完成確認。

5.3 結語

目前一般社會大眾，對於租借單車普遍熟悉的方式，還是偏向傳統的固定站點模式租借，但隨著 ofo 與摩拜單車在中國的出現，同時也慢慢開始使用這種無站點式的租借服務。然而，儘管無站點式的租借模式看似比起固定站點式來得更具便利性，但卻也引發了其他基於使用者公德心的爭議問題，像是車輛違停，或是私佔車輛等。

或許有人認為大部分現有市場上出現的無站點式單車租借服務，都不是真正的單車共享概念，原因除了車輛的來源都是營運商自行生產或購買，而非利用客戶端手上的閒置資源以外，還有就是這種由營運商統一提供車輛的無站點式單車租借服務，在未達到足夠規模化之前，可能因為使用者貪圖自身便利，把單車當作自己的專屬車輛，發生藏匿占據的情況，導致更容易被當成一種長期租用的個人單車，反而少了 Ubike 那種依靠限制構成的「共享」。

要能真正落實單車共享的概念，除了需要具備良好的公民道德素養以外，都市的規劃也是很重要的一環，這其中包括單車專用道的設置，以及滿足供需平衡的合法單車停車位等友善於騎乘者的環境，都是有利於推動單車共享的因素之一。

本篇研究所提出的智能合約單車租借，希望利用近年蓬勃發展的區塊鏈技術，結合智能合約的可能性，創造一個能夠提供單車共享的租借平台工具。智能合約單車租借的概念，或許不是一個完善的商業模式，或是一個可行的獲利方法，仍有一些軟硬體方面的缺失需要重新設計與整合，但卻不可否認是一個實現單車共享經濟的方式之一，也期盼後續研究將能為智能合約單車租借提出更明確有效的運行系統。

參考文獻

英文文獻

1. Bentov, A. Gabizon, and A. Mizrahi, \Cryptocurrencies without proof of work," CoRR, vol. abs/1406.5694, 2014.
2. J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, “Sok: Research perspectives and challenges for bitcoin and cryptocurrencies,” in IEEE Symposium on Security and Privacy, pp. 104–121, May 2015.
3. N. T. Courtois and L. Bahack, “On subversive miner strategies and block withholding attack in bitcoin digital currency,” CoRR, vol. abs/1402.1718, 2014.
4. Eyal and E. G. Sirer, “Majority is not enough: Bitcoin mining is vulnerable,” CoRR, vol. abs/1311.0243, 2013.
5. J. Garay, A. Kiayias, and N. Leonardos, The Bitcoin Backbone Protocol: Analysis and Applications, pp. 281–310, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
6. Gervais, G. O. Karame, V. Capkun, and S. Capkun, “Is bitcoin a decentralized currency?,” IEEE Security Privacy, vol. 12, pp. 54–60, May 2014.
7. A. Gervais, G. O. Karame, K. W¨ust, V. Glykantzis, H. Ritzdorf, and S. Capkun, “On the security and performance of proof of work blockchains,” in Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS’16), pp. 3–16, New York, NY, USA, 2016.
8. A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, “Tampering with the delivery of blocks and transactions in bitcoin,” in Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security (CCS’15), pp. 692–705, New York, NY, USA, 2015.
9. E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, “Eclipse attacks on bitcoin’s peer-to-peer network,” in 24th USENIX Security Symposium, pp. 129–144, Washington, D.C., 2015.
10. G. Karame, “On the security and scalability of bitcoin’s blockchain,” in Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS’16), pp. 1861–1862, New York, NY, USA, 2016.
11. G. O. Karame, “Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin,” in Proceedings of Conference on Computer and Communication Security, pp. 1–17, 2012.
12. S. King and S. Nadal, Ppcoin: Peer-to-peer Crypto-Currency with Proof-of-Stake, 2012. (https://archive.org/stream/PPCoinPaper/ppcoin-paper_djvu.txt)

13. A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, “Hawk: The blockchain model of cryptography and privacy-preserving smart contracts,” in 2016 IEEE Symposium on Security and Privacy (SP’16), pp. 839–858, May 2016.
14. L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, “A secure sharding protocol for open blockchains,” in Proceedings of ACM SIGSAC Conference on Computer and Communications Security (CCS’16), pp. 17–30, New York, NY, USA, 2016.
15. S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, Feb. 24, 2013. (http://bitcoin.org/bitcoin.pdf)
16. M. Rosenfeld, “Analysis of hashrate-based double spending,” CoRR, vol. abs/1402.2009, 2014.
17. Y. Sompolinsky and A. Zohar, Secure High-Rate Transaction Processing in Bitcoin, pp. 507–527, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
18. W. T. Tsai, R. Blower, Y. Zhu, and L. Yu, “A system view of financial blockchains,” in IEEE Symposium on Service-Oriented System Engineering (SOSE’16), pp. 450–457, Mar. 2016.
19. H. Watanabe, S. Fujimura, A. Nakadaira, Y. Miyazaki, A. Akutsu, and J. Kishigami, “Blockchain contract: Securing a blockchain applied to smart contracts,” in IEEE International Conference on Consumer Electronics (ICCE’16), pp. 467–468, Jan. 2016.