中華民國第 57 屆中小學科學展覽會
作品說明書

高級中等學校組　電腦與資訊學科

佳作

最佳創意獎

052505

使用區塊鏈開發可監督捐款之公開募捐平台

學校名稱：國立臺南第一高級中學

作者：	指導老師：
高三 蘇俊嘉 高二 陳昱丞 高二 周奕寬	高英耀

關鍵詞：BlockChain、Smart Contract、公益募捐

摘要

新興的 BlockChain（區塊鏈）技術具有去中心化及資料無法被竄改的特性，在需要資料透明的領域非常適用，尤其 Fintech（金融科技）方面因貨幣貴重的特性而特別有優勢。慈善募款常常有用款不透明與部分用途不被大眾認可的問題；因此本研究使用 BlockChain 技術之 Ethereum（以太坊）平台，成功設計出一套 Smart Contract（智能合約），主要功能針對募款組織的所有撥款，須經捐款者投票，通過後才可撥用，將監督的權利歸還給捐款者，使慈善捐款能夠公開透明且受監督。同時也研究網頁前後端程式語言，設計一套大眾能簡便進行操作之介面，期望真正將此募捐平台推廣至社會。

壹、研究動機

臺灣公益責信協會於 2015 年統計，全臺六百大非營利組織基金會，有高達八成未提供財務報告，尤其許多慈善機構不常公開捐款用途；近年也有新聞報導指出，有些慈善機構將民眾善款用於非公益事項；層出不窮的事件引起我們反思：難道捐款者沒有辦法知道自己的善心究竟被用到哪了嗎？莫非捐款者沒有監督捐款用途的權利嗎？

貳、研究目的

為了改善上述問題，故希望透過區塊鏈技術開發出可監督捐款的公開募捐平台，並期望達到以下幾點。

一、公開且無法竄改：利用 BlockChain 使捐款用途公開化且資料無法被竄改

二、用戶的監督權利：藉由 Ethereum Smart Contract 達成令捐款者擁有投票權監督所有用款

三、簡便的操作介面：研究網頁前後端程式語言，設計出簡易的 UI（使用者介面）

四、改善責信公信力：提升公益責信度，讓社會大眾能夠更放心得捐款

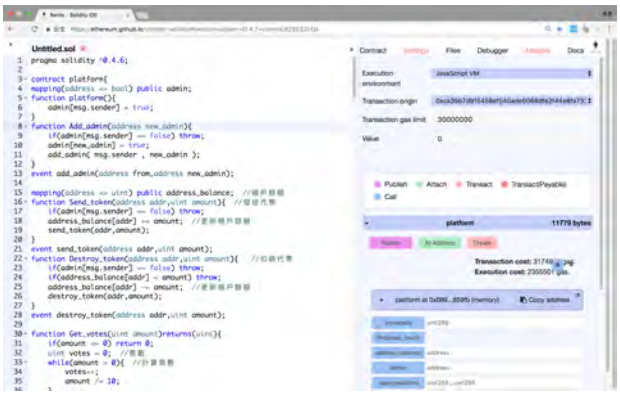
參、研究設備及器材

一、硬體設備

紙、筆、電腦、伺服器

二、軟體

Remix、Ethereum Mist、Sublime Text

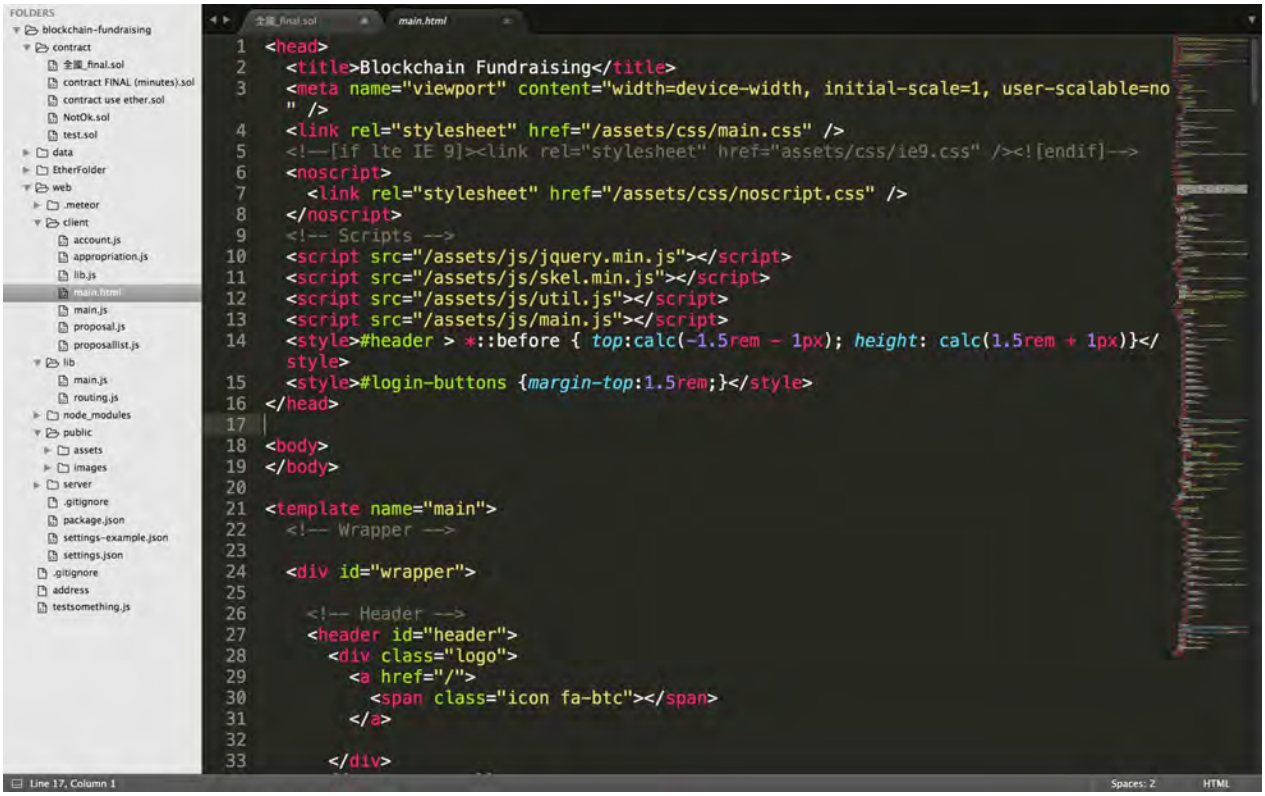


Remix 是基於瀏覽器的 Solidity 編譯器和 IDE

Mist 是 Ethereum 平台上用來瀏覽鏈上資料和使用 BlockChain 的工具之一

圖 3.1：Remix

圖 3.2：Ethereum Mist



Sublime Text 是一套跨平台的文字編輯器

圖 3.3：Sublime Text

肆、研究過程或方法

一、募捐平台研究步驟圖

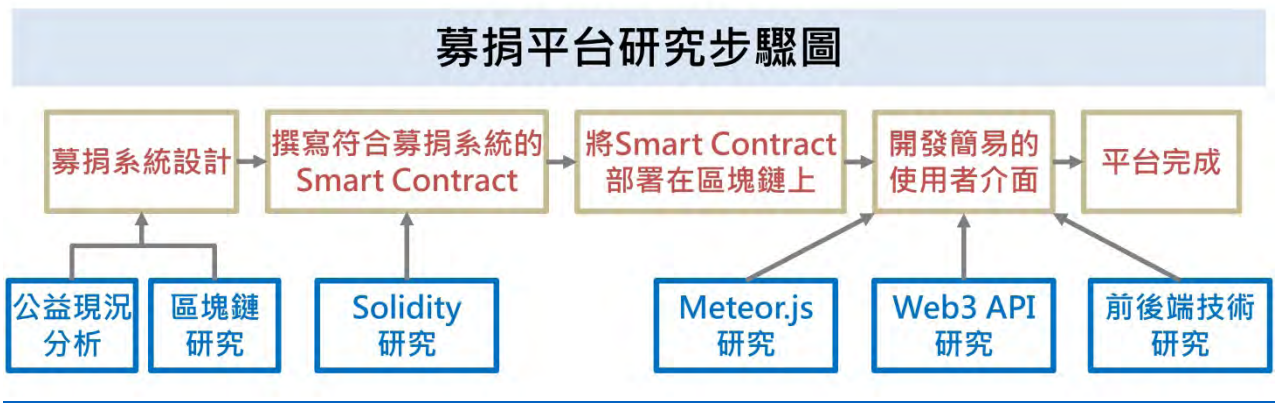


圖 4.1：募捐平台研究步驟圖

二、基本名詞的介紹

(一) BlockChain（區塊鏈）

BlockChain（區塊鏈）是一個透過「去中心化」和「去信任化」的方式來共同維護資料庫的技術，其將資料庫分別放在不同的節點裡保存，並互相監控數據，資料更動須其餘節點的共識同意。它讓交易過程中每個節點的每一筆交易，都能透明、安全地被紀錄下來。而各交易被廣播到全網後一段時間，會被收進一個區塊（Block），和全部的節點複製共享；隨著時間區塊一個一個生成形成一條鏈（Chain）。而所有節點都會擁有所有交易紀錄，共同驗證鏈上資料的正確性，故極難利用少數節點竄改歷史數據。

(二) Bitcoin（比特幣）與 Ethereum（以太坊）

Bitcoin 是一種基於 BlockChain 的數位貨幣，安全、匿名且無國界之分。其去中心化的特性，解決了傳統交易須藉由第三方解決彼此的信賴問題。所有交易由眾多節點共同去驗證，人人都可以參與。利益因素則降低了節點作惡的可能。

Ethereum 是一個 BlockChain 應用平台，藉由 Solidity 語言編寫 Smart Contract 並於 Ethereum Virtual Machine（EVM，以太坊虛擬機）上執行，允許使用者在平台上創建 BlockChain 應用，且應用範疇不僅限於數位貨幣。

(三) Smart Contract（智能合約）

一個 Smart Contract 是一套以數位形式定義的承諾。Ethereum Smart Contract 是運行在 BlockChain 上的程式，將條件和結果寫成程式碼，每個用戶皆可對 Smart Contract 發送交易，而合約上約定好的事情無法竄改，使得所有在合約上的**計算變得可信任**。

(四) Solidity 語言

Solidity 是一種用來編寫 Smart Contract 的程式語言，它的語法接近於 Javascript，是一種物件導向的語言。

三、募捐系統設計

(一) 募捐系統概念設計

1. **用戶**：平台中有一般用戶與管理員兩種，管理員具有公正交換法幣與代幣之權限。
2. **募款**：而每一個用戶皆可進行依其目的或機構名稱進行募款，例如「xx 慈善機構」即為一種募款，「幫助地震災民」也是一種。在募款時應提供發起撥款時投票反對門檻與可投票時間供用戶決定是否捐款。而每個募款應要填寫：
   (1)募款名稱 - 說明募款大綱
   (2)募款內容 - 詳細說明募款細項
   (3)**反對門檻** - 為一正整數的百分比，若總反對票數佔總票數的百分比超過此反對門檻，則反對成功。
   (4)募款中每個撥款的**可投票時間**（小時） - 讓每個捐款者對撥款的可投票時間
   ※反對門檻與可投票時間須且僅能在募款發起時就設定好
3. **捐款**：在每一個募款之中，皆可以接受所有用戶的捐款，有捐款至募款的用戶，便具有監督該募款使用款項的權利。
4. **撥款**：每一個**募款發起者**皆可以在其募款中依其募款所募得之善款進行多次撥款。而每個撥款應要填寫：
   (1)撥款名稱 - 說明撥款大綱
   (2)撥款內容 - 詳細說明撥款細項
   (3)撥款金額 - 將要從該募款撥出多少款項
   (4)受助者 - 撥款成功時，款項送往的目標。**受助者可為一個帳戶或一個募款**

5. **投票監督**：在發起該撥款後，該時刻前所有有向該募款捐款的人，在可投票時間內都有向該撥款投反對票的權利。若某個用戶在該募款中的捐款金額為**正整數** D，則該捐款金額可獲得之票數為

$$floor(\log_{10}(D))+1$$

即 D 以 10 為底取對數向下取整再加 1（表 3-1，可得知票數對應金額的成長是越趨緩慢的，換言之，一人想獲得絕大多的票是極難的）。若反對總票數的在總票數的百分比超過了反對門檻則不會順利撥款，否則會順利撥款至受益者。

表 4-1：**捐款金額 D 與票數計算公式的簡易對照表**

捐款區間	[1,10)	[10,10^2)	[10^2,10^3)	[10^3,10^4)	[10^4,10^5)	[10^5,10^6)	[10^6,10^7)
對應票數	1	2	3	4	5	6	7

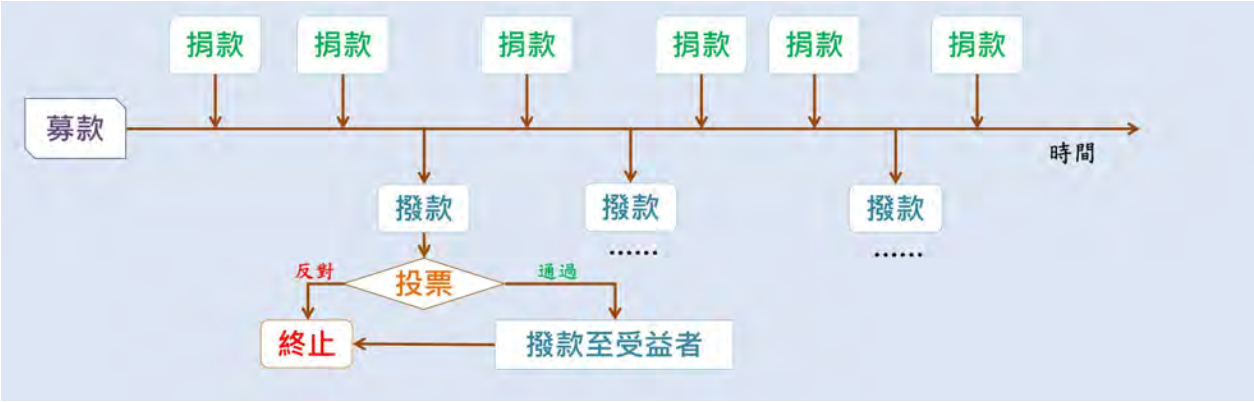


圖 4.2：募捐系統設計架構圖

(二) 與實務串接

若要利用法幣做捐款，而**現行的法幣都還不是數位貨幣的形式**，導致 BlockChain 上記載的代幣僅是代表法幣的資訊，故必須有個**可信任的第三方機構**來做代幣與法幣 **1:1 的交換**，。且須要做身份認證，達到**一人一帳戶**而管理員為可信任的第三方機構，例如：銀行、政府機關等等。

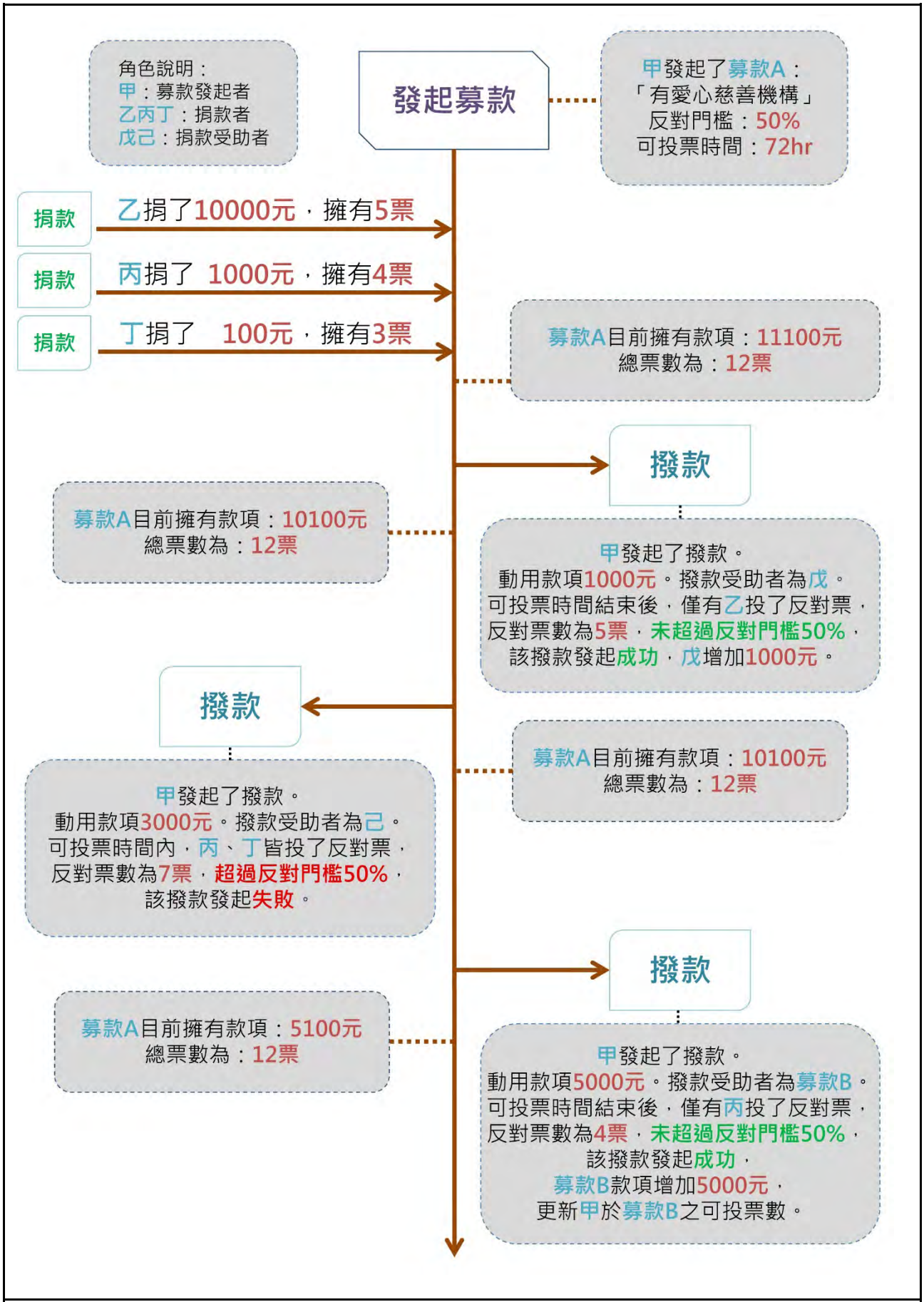

角色說明：
甲：募款發起者
乙丙丁：捐款者
戊己：捐款受助者
發起募款
甲發起了募款A：
「有愛心慈善機構」
反對門檻：50%
可投票時間：72hr
捐款
乙捐了10000元，擁有5票
捐款
丙捐了 1000元，擁有4票
捐款
丁捐了 100元，擁有3票
募款A目前擁有款項：11100元
總票數為：12票
撥款
募款A目前擁有款項：10100元
總票數為：12票
甲發起了撥款。
動用款項1000元。撥款受助者為戊。
可投票時間結束後，僅有乙投了反對票，
反對票數為5票，未超過反對門檻50%，
該撥款發起成功，戊增加1000元。
撥款
募款A目前擁有款項：10100元
總票數為：12票
甲發起了撥款。
動用款項3000元。撥款受助者為己。
可投票時間內，丙、丁皆投了反對票，
反對票數為7票，超過反對門檻50%，
該撥款發起失敗。
撥款
募款A目前擁有款項：5100元
總票數為：12票
甲發起了撥款。
動用款項5000元。撥款受助者為募款B。
可投票時間結束後，僅有丙投了反對票，
反對票數為4票，未超過反對門檻50%，
該撥款發起成功，
募款B款項增加5000元，
更新甲於募款B之可投票數。

圖 4.3：募捐實例圖

四、Smart Contract 流程圖

我們先將整個募捐系統架構所需的邏輯寫成流程圖。

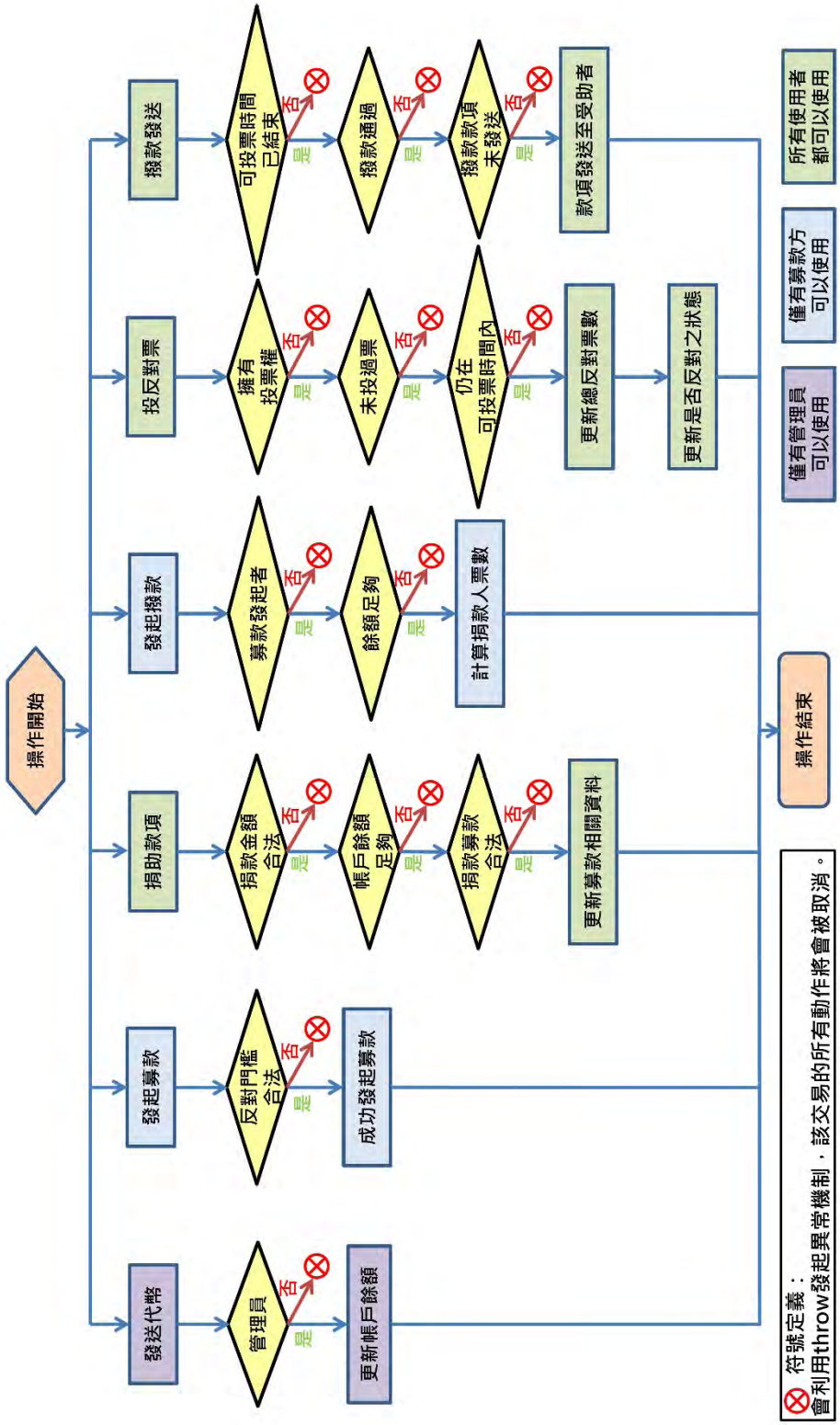


圖 4.5： Smart Contract 流程圖

五、Smart Contract 研究開發

利用 Solidity 語言將「四、Smart Contract 流程圖」的設計編寫成 Smart Contract，以下〔表 4-2〕對於 Solidity 語法做簡單的介紹。

名詞	名詞解釋
地址(address)	每一個使用者都會存放於一個地址。
事件(event)	事件內容會被記錄在 BlockChain 上。
映射(mapping)	將一種型別資料對應至另一種型別資料的資料結構。
msg.sender	將會回傳發送該交易的帳戶地址。

表 4-2：Solodity 語法解釋

(一) 定義平台的資料結構及設置管理員權限

1. 第 4 行：admin：帳戶地址映射到其是否為管理員。
2. 第 5~7 行：該函式只會在發起該平台合約時呼叫，並將平台發起人設為管理員。
3. 第 8~12 行：Add_admin 函式，用來將目標帳戶地址設置為管理員。
4. 第 8 行：傳入 *new_admin* 參數，表示要設定為管理員的帳戶地址。
5. 第 9 行：限定只有管理員可以呼叫此函式。

```
3  contract platform{
4  mapping(address => bool) public admin;
5  function platform(){
6      admin[msg.sender] = true;
7  }
8  function Add_admin(address new_admin){
9      if(admin[msg.sender] == false) throw;
10     admin[new_admin] = true;
11     add_admin( msg.sender , new_admin );
12 }
13 event add_admin(address from,address new_admin);
```

程式碼片段 4.1：Admin 區塊程式碼

(二) 設置代幣(Token)

1. 第 16~20 行：Send_token 函式，用來發送代幣。

2. 第 16 行：傳入參數 *addr* 為代幣送往的帳戶地址、*amount* 為發送代幣數。

3. 第 17 行：限定只有管理員可以呼叫此函式。

4. 第 22~27 行： Destroy_token 函式，用來扣除代幣。

5. 第 22 行：傳入參數 *addr* 為要扣除代幣的帳戶地址、*amount* 為扣除代幣數。

6. 第 23 行：限定只有管理員可以呼叫此函式。

7. 第 24 行：確認該帳戶地址是否有足夠的代幣。

```
15 mapping(address => uint) public address_balance;  //帳戶餘額
16 function Send_token(address addr,uint amount){  //發送代幣
17     if(admin[msg.sender] == false) throw;
18     address_balance[addr] += amount;  //更新帳戶餘額
19     send_token(addr,amount);
20 }
21 event send_token(address addr,uint amount);
22 function Destroy_token(address addr,uint amount){   //扣除代幣
23     if(admin[msg.sender] == false) throw;
24     if(address_balance[addr] < amount) throw;
25     address_balance[addr] -= amount;  //更新帳戶餘額
26     destroy_token(addr,amount);
27 }
28 event destroy_token(address addr,uint amount);
```

程式碼片段 4.2：Token 區塊程式碼

(三) 捐款與票數的轉換

1. 第 30~38 行：Get_votes 函式，用來根據捐款者捐款額計算其所擁有的票數。

2. 第 30 行：傳入參數 *amount* 為捐款金額，而回傳的是其對應到的票數。

3. 第 31 行：無捐款，票數為零。

4. 第 33~36 行：捐款金額以 10 為底取對數向下取整再加 1，即為捐款金額所對應的票數。

```
30 function Get_votes(uint amount)returns(uint){
31     if(amount <= 0) return 0;
32     uint votes = 0;  //票數
33     while(amount > 0){  //計算票數
34         votes++;
35         amount /= 10;
36     }
37     return votes;  //回傳票數
38 }
```

程式碼片段 4.3：Get_votes 函式

(四) 募款結構

1. 第 41 行：FRs，募款編號映射到相對應的募款。

2. 第 42 行：Donation：募款編號與捐款者帳戶地址映射到，該募款下捐款者的編號。

3. 第 43 行：AddrList：募款編號與捐款者編號映射到，其於該募款之捐款總額。

4. 第 44~57 行：Fundraising：募款結構，包含各種募款相關資訊。

```
40 uint public Fundraising_count;  //總募款數
41 mapping(uint=>Fundraising ) public FRs;
42 mapping(uint=>mapping(address=>uint)) public Donation;
43 mapping(uint=>mapping(uint=>address)) public AddrList;
44 struct Fundraising{
45     address Founder;  //募款受助者
46     string Name;  //募款名稱
47     uint FR_id;  //募款編號
48     uint TotalDonations;  //總捐款數
49     uint balance;  //剩餘捐款數
50     uint people_count;  //捐款人數
51     uint Totalvotes;  //總票數
52     uint begin_time;  //發起募款時間
53     uint VoteTime;  //撥款的可投票時間
54     uint BANpercentage;  //撥款的反對門檻
55     uint App_count;  //總撥款數
56     string INFO;  //募款內容
57 }
```

程式碼片段 4.4：Fundraising 資料結構

(五) 發起募款

1. 第 59~81 行：Add_Fundraising 函式，用來發起一個新的募款。

2. 第 60 行：傳入參數 *_Name* 為募款名稱、 *_BANpercentage* 為反對撥款所需要超過的總票數門檻百分比、*_Vote_time* 為該募款下所有撥款的可投票時間、*_info* 為募款的內容。

3. 第 64 行：反對門檻為 0%，表示只要有人投反對票，該撥款就會被反對，因為如果捐款者數量多且金額多，有可能一個人擁有的票數不到總可投票數的 1%，故 1% 不表示有人投票就會被反對，而 0% 才是。

4. 第 65~78 行：設定募款相關資料。

```
59 function Add_Fundraising(  //發起募款
60     string _Name,uint _BANpercentage,uint _VoteTime,string _info)
61     returns(uint FR_id){
62     Fundraising_count++;  //更新總募款數
63     uint time_tp = now;  //now回傳當前時間
64     if(_BANpercentage<0||_BANpercentage>100) throw;//非法的反對門檻
65     FRs[Fundraising_count] = Fundraising({
66         Founder : msg.sender,
67         Name : _Name,
68         FR_id : Fundraising_count,
69         TotalDonations : 0,
70         people_count : 0,
71         balance : 0,
72         Totalvotes : 0,
73         begin_time : time_tp,
74         VoteTime : _VoteTime,
75         BANpercentage : _BANpercentage,
76         App_count : 0,
77         INFO : _info
78     });
79     add_Fundraising(msg.sender,_Name,Fundraising_count,_info);
80     return Fundraising_count;
81 }
82 event add_Fundraising(
83     address owner_address,
84     string Fundraising_Name,
85     uint Fundraising_id,
86     string INFO
87 );
```

程式碼片段 4.5：Add_Fundraising 函式

(六) 撥款結構

1. 第 89 行：Apps：是由募款編號與撥款編號映射到對應的撥款，例如 Apps[2][5] 就是代表編號 2 募款的編號 5 撥款。

2. 第 90~91 行：AtDonation：由募款編號、撥款編號與帳戶地址映射捐款者在該撥款發起時之總款總額。

3. 第 92~93 行：VoteDone：由募款編號、撥款編號與帳戶地址映射捐款者是否已對該撥款投過票。

4. 第 94~107 行：撥款結構。包含各種撥款相關資訊。

```
89  mapping(uint=>mapping(uint=>Appropriation)) public Apps;
90  mapping(uint=>mapping(uint=>
91      mapping(address=> uint))) public AtDonation;
92  mapping(uint=>mapping(uint=>
93      mapping(address=> bool))) public VoteDone;
94  struct Appropriation{
95      string Name;  //撥款名稱
96      uint FR_id;  //募款編號
97      uint Appropriation_id;  //撥款編號
98      address recipient;  // 受助者(帳戶)
99      uint recipient_FRid;  //受助者(募款)
100     uint Token_Amount;  //撥款所須之代幣數
101     uint BAN_votes;  //當前反對票數
102     uint Totalvotes;  //總可投票數
103     uint begin_time;  //撥款開始時間
104     bool BAN;  //該撥款是否已被反對
105     bool paid_token;  //是否已撥款
106     string INFO;  //撥款內容
107 }
```

程式碼片段 4.6：Appropriation 資料結構

(七) 發起撥款（撥款受助者為帳戶）

1. 第 109~139 行：Add_Appropriation 函式。用來發起一個撥款。

2. 第 109~110 行：傳入參數 *_FR_id* 為募款編號、*_Name* 為撥款名稱、*_amount* 為撥款金額、*_recipient* 為受助者的帳戶地址、*info* 為撥款內容。

3. 第 116~129 行：設定撥款相關資料。

4. 第 124 行：總可投票數為撥款發起時，募款當前的總可投票數。

5. 第 132~136 行：紀錄撥款發起時，每個捐款者的捐款金額；用來之後計算票數。

```
109 function Add_Appropriation(uint _FR_id ,  //撥款給帳戶地址
110     string _Name,uint _amount,address _recipient,string _info){
111     Fundraising FR = FRs[_FR_id];
112     if(msg.sender != FR.Founder) throw; //撥款發起者非募款發起者
113     if(FR.balance < _amount) throw;  //撥款所需餘額不足
114     uint time_tp = now;  //now回傳當前時間
115     FR.App_count++;  //更新該募款之總撥款數
116     Apps[_FR_id][FR.App_count] = Appropriation({
117         Name : _Name ,
118         FR_id : _FR_id ,
119         Appropriation_id : FR.App_count ,
120         recipient : _recipient ,  //撥款受助者為帳戶
121         recipient_FRid : 0x0 ,
122         Token_Amount : _amount ,
123         BAN_votes : 0 ,
124         Totalvotes : FR.Totalvotes ,  //總可投票數
125         begin_time : time_tp ,
126         BAN : false ,
127         paid_token : false,
128         INFO: _info
129     });
130     FR.balance -= _amount; //從募款的餘額中扣除撥款所需之代幣數
131     Appropriation App = Apps[_FR_id][FR.App_count];
132     for(uint i = 1 ; i <= FR.people_count ; i++){
133         address addr = AddrList[_FR_id][i] ;
134         AtDonation[_FR_id][FR.App_count][addr] =
135             Donation[_FR_id][addr];
136     }
137     add_appropriation(msg.sender,App.recipient,
138         FR.Name,_FR_id,_Name,FR.App_count,_info);
139 }
140 event add_appropriation(
141     address owner_address,
142     address recipient_address,
143     string Fundraising_name,
144     uint Fundraising_id,
145     string appropriation_name,
146     uint appropriation_id,
147     string INFO
148 );
```

程式碼片段 4.7：Add_Appropriation 函式

(八) 發起撥款（撥款受助者為募款）

1. 第 150~178 行：Add_Appropriation_To_FR 函式。用來發起受助者為募款的撥款。

2. 第 151~152 行：傳入參數 *_FR_id* 為募款編號、*_Name* 為撥款名稱、*_amount* 為撥款金額、*_recipient_PROid* 為受助者募款編號、*_info* 為撥款內容。

3. 第 159~172 行：設定撥款相關資料。

4. 第 167 行：總可投票數為撥款發起時，募款當前的總可投票數。

5. 第 174~178 行：紀錄撥款發起時，募款下每個捐款者之捐款金額。

```
150 function Add_Appropriation_To_FR(  //撥款給其他募款
151     uint _FR_id ,string _Name,uint _amount,
152     uint _recipient_FRid,string _info){
153     Fundraising FR = FRs[_FR_id];
154     if(msg.sender != FR.Founder) throw;  //撥款發起者非募款發起者
155     if(FR.balance < _amount) throw;  //撥款所須餘額不足
156     if(_FR_id == _recipient_FRid ) throw; //受助募款為原本募款
157     uint time_tp = now;  //now回傳當前時間
158     FR.App_count++;  //更新該募款之總撥款數
159     Apps[_FR_id][FR.App_count] = Appropriation({
160         Name : _Name,
161         FR_id : _FR_id,
162         Appropriation_id : FR.App_count,
163         recipient : 0x0,
164         recipient_FRid : _recipient_FRid, //撥款受助者為募款
165         Token_Amount : _amount,
166         BAN_votes : 0,
167         Totalvotes : FR.Totalvotes,  //總可投票數
168         begin_time : time_tp,
169         BAN : false,
170         paid_token : false,
171         INFO: _info
172     });
173     FR.balance -= _amount;  //從募款的餘額中扣除撥款所需之代幣數
174     for(uint i=1 ; i <= FR.people_count ; i++){
175         address addr = AddrList[_FR_id][i] ;
176         AtDonation[_FR_id][FR.App_count][addr] =
177             Donation[_FR_id][addr];
178     }
179     add_appropriation_To_FR(msg.sender,_recipient_FRid,
180         FR.Name,_FR_id,_Name,FR.App_count,_info);
181 }
182 event add_appropriation_To_FR(
183     address owner_address,
184     uint recipient_FRid,
185     string Fundraising_name,
186     uint Fundraising_id,
187     string appropriation_name,
188     uint appropriation_id,
189     string INFO
190 );
```

程式碼片段 4.8：Add_Appropriation_To_FR 函式

(九) 捐款

1. 第 192~219 行：Donate 函式，用來捐款給募款。

2. 第 190 行：傳入參數 *FR_id* 為捐款之目標募款編號、*amount* 為捐款金額。

3. 第 194 行：捐款金額不足。

4. 第 195 行：目標捐款募款不存在。

5. 第 197 行：捐款者為初次捐款至該募款。

6. 第 199 行：設定捐款者編號。

7. 第 201~204 行：非第一次捐款，更新募款之總可投票數(第 206~207 行)的前置作業。

8. 第 205 行：更新捐款者於該募款之捐款總額。

9. 第 206~207 行： 更新募款之總可投票數

10. 第 209 行：更新募款之總捐款金額。

11 第 210 行：更新募款之餘額。

```
192 function Donate(uint FR_id,uint amount){
193     if(amount <= 0) throw;  //捐款數不合法
194     if(address_balance[msg.sender] < amount) throw;
195     if(FRs[FR_id].Founder == 0x0 ) throw;
196     Fundraising FR = FRs[FR_id];
197     if(Donation[FR_id][msg.sender] == 0){
198         FR.people_count++;  //該募款總捐款人數增加
199         AddrList[FR_id][FR.people_count] = msg.sender;
200     }
201     else if(Donation[FR_id][msg.sender] > 0){
202         FR.Totalvotes -= Get_votes(
203             Donation[FR_id][msg.sender]);
204     }
205     Donation[FR_id][msg.sender] += amount;
206     FR.Totalvotes +=
207         Get_votes(Donation[FR_id][msg.sender]);
208     address_balance[msg.sender] -= amount;
209     FR.TotalDonations += amount;
210     FR.balance += amount;
211     donate(msg.sender,amount,FR.Name,FR_id,"Account");
212 }
213 event donate(
214     address From,
215     uint amount,
216     string to_Fundraising_name,
217     uint to_FR_id,
218     string FromWho
219 );
```

程式碼片段 4.9：Donate 函式

(十) 投票

1. 第 221~238 行：Vote_BAN 函式，用來對撥款投反對票。
2. 第 221 行：傳入參數 *FR_id* 為募款編號、*App_id* 為撥款編號。
3. 第 224 行：該帳戶於撥款發起前無捐款紀錄，無投票權。
4. 第 225 行：該帳戶已經投過票了。
5. 第 226~227 行：未在可投票時間內。
6. 第 229 行：將帳戶標記為已投票。
7. 第 230~231 行：更新撥款之總反對票。
8. 第 232 行：該撥款已被反對，退出函式。
9. 第 233 行：反對票數超過反對門檻。
10. 第 234 行：款項發送回原本的募款。
11. 第 235 行：將撥款標記為已被反對。避免之後款項重複流回募款。

(※於可投票時間內，即便撥款已被反對，仍然能投票)

```
221 function Vote_BAN(uint FR_id,uint App_id){  //投票
222     Appropriation App = Apps[FR_id][App_id];
223     Fundraising FR = FRs[App.FR_id];
224     if(AtDonation[FR_id][App_id][msg.sender] == 0) throw;
225     if(VoteDone[FR_id][App_id][msg.sender] == true) throw;
226     if(now > App.begin_time
227         + FR.VoteTime * 1 hours) throw;
228
229     VoteDone[FR_id][App_id][msg.sender] = true;
230     App.BAN_votes += Get_votes(
231         AtDonation[FR_id][App_id][msg.sender]);
232     if(App.BAN == true) return ;
233     if(App.BAN_votes*100 >= App.Totalvotes*FR.BANpercentage){
234         FR.balance+=App.Token_Amount;
235         App.BAN = true;
236     }
237     vote_ban(FR_id,App_id,msg.sender);
238 }
239 event vote_ban(uint FR_id,uint App_id,address send_address);
```

程式碼片段 4.10：Vote_BAN 函式

(十一) 檢查並進行代幣發放

1. 第 241~272 行：Check_pass 函式，用來檢查撥款是否通過，並發送代幣至受助者。
2. 第 241 行：傳入參數 *FR_id* 為募款編號、*App_id* 為撥款編號。
3. 第 250 行：發送代幣至目標帳戶。
4. 第 256 行：募款發起人為第一次捐款給受助者募款。
5. 第 258 行：設定募款發起人於受助者募款之捐款者編號。
6. 第 260 行：募款發起人非第一次捐款給受助者募款。
7. 第 261~262 行：更新受助者募款之總可投票數(第 265 行)的前置作業。
8. 第 264 行：更新募款發起人於受助者募款之捐款總額。
9. 第 265 行：更新受助者募款之總可投票數。

```
241 function Check_pass(uint FR_id,uint App_id){//確認撥款是否通過
242     Appropriation App = Apps[FR_id][App_id];
243     Fundraising FR = FRs[FR_id];
244     if(now < App.begin_time +
245         FR.VoteTime * 1 hours) throw;  //投票尚未結束
246     if(App.BAN == true) throw;  //撥款已終止
247     if(App.paid_token == true) throw;  //撥款款項已發送
248
249     if(App.recipient!=0x0){  //受助者是帳戶
250         address_balance[App.recipient] += App.Token_Amount;
251     }
252     else{  //受助者是募款
253         uint amount = App.Token_Amount;
254         Fundraising TO_FR = FRs[App.recipient_FRid];
255         uint ID = TO_FR.FR_id;
256         if(Donation[ID][FR.Founder] == 0){
257             TO_FR.people_count++;  //受助募款總捐款人數增加
258             AddrList[ID][TO_FR.people_count]=FR.Founder;
259         }
260         else if(Donation[ID][FR.Founder] > 0){
261             TO_FR.Totalvotes
262                 -=Get_votes(Donation[ID][FR.Founder]);
263         }
264         Donation[ID][FR.Founder] += amount;
265         TO_FR.Totalvotes +=Get_votes(Donation[ID][FR.Founder]);
266         TO_FR.TotalDonations += amount;  //更新捐款總額
267         TO_FR.balance += amount;  //更新捐款餘額
268         donate(FR.Founder,amount,TO_FR.Name,ID,"Fundraising");
269     }
270     App.paid_token = true;  //撥款款項標記為已發送
271     check_pass(FR_id,App_id,msg.sender);
272 }
273 event check_pass(uint FR_id,uint App_id,address send_address);
```

程式碼片段 4.11：Check_pass 函式

六、利用 Remix 與 Mist 進行 Smart Contract 測試並部署至 Ethereum

Remix 是基於瀏覽器的 Solidity 編譯器和 IDE，Mist 是 Ethereum 平台上用來瀏覽鏈上資料和使用 BlockChain 的工具之一。我們將平台的 Smart Contract 利用 Rimix 進行測試與除錯，最後我們利用 Mist 再將其部署在 Ethereum 上，並確認了系統的可實行性。下圖可看到為將 Smart Contract 部署在 Ethereum 上的交易紀錄。

圖 4.6：Ethereum 部署 Smart Contrac 的交易紀錄

七、開發簡便介面

為了讓使用者不需要使用較難操作的底層操作方式，我們使用了 Meteor.js 框架開發簡易 UI(使用者介面)，再利用後端的 web3 API 讀取 BlockChain 上之資料，進行整理之後呈現 在網頁上。以下為我們的 UI 架構圖與一些片段展示。

註：

Web3 API (Application Programming Interface) ：Web3 包含了一個 eth 的物件，Web3.eth 是用來與 Ethereum BlockChain 溝通，可讀取將 Ethereum 上的資料與發送交易。

Template： Template 可以用來表達一個頁面的架構

BlazeLayout ：在此主要用來 render 出指定的 Template，常與 FlowRouter 做搭配使用

FlowRouter ：在此主要用來將每個頁面設定出一個連結路徑

HTML5-UP：提供很多網站模板，方便我們快速用修改的方式架設網頁。

Meteor.js：Meteor.js 為結合前後端的 JavaScrip 框架，後端伺服器運行於 Node.js，資料庫連結 MongoDB，使用 JavaScript 單一語言就能完成前端網頁呈現和後端伺服器架網頁。

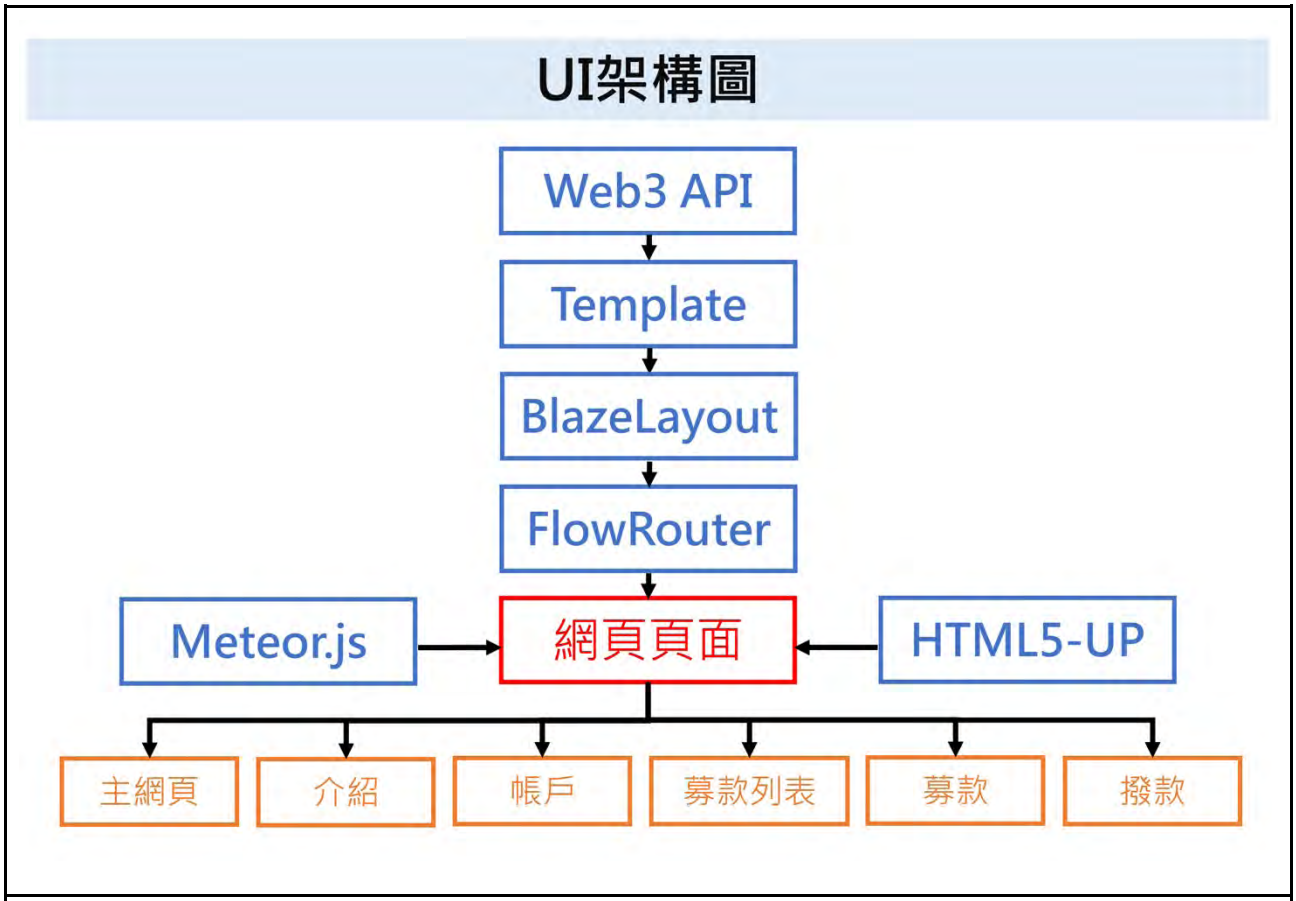


圖 4.7：UI 設計架構圖

```
79  <template name="account">
80    <article id="account">
81      <h1 class="major">我的帳戶</h1>
82      {{#if guest_myaccount}}
83        你必須先登入!!
84      {{else}}
85        <section>
86          <div style="text-align:center; margin:0px auto;">
87            帳戶地址: {{ID}} <br>
88            帳戶餘額: {{balance}} <br>
89          </div>
90        </section>
91        {{#if myaccount}}
92          <button id="add">購買代幣</button>
93          <button id="TakeMoney">兌換法幣</button>
94        {{/if}}
95      {{/if}}
96      <br><br>
97      <h2>我發起的募款列表</h2>
98      <ul>
99        {{#each list}}
100         <li><a href="/proposal/{{args.proposal_id}}">
            {{args.proposal_Name}}</a></li>
101       {{/each}}
102     </ul>
103     {{#if myaccount}}
104       <h2>我捐過的募款:</h2>
105       <ul>
106         {{#each donatedproposal}}
107           <li> <a href="/proposal/{{args.to_proposal_id}}">
              {{args.to_proposal_name}}</a>: {{args.amount}}NTD</li>
108         {{/each}}
109       </ul>
110     {{/if}}
111   </article>
112 </template>
```

圖 4.8：UI 片段程式碼(1)

```
22 FlowRouter.route('/account/:id', {
23   name: 'account',
24   action() {
25     BlazeLayout.render('main', {content: 'account'});
26   }
27 });
```

圖 4.9：UI 片段程式碼(2)

伍、研究結果

一、藉由 Smart Contract 與 Ethereum BlockChain ，設計了一套有以下特點的募捐平台。

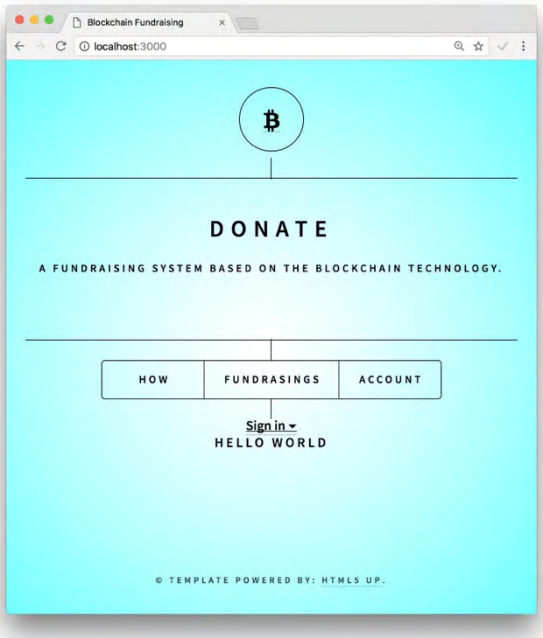
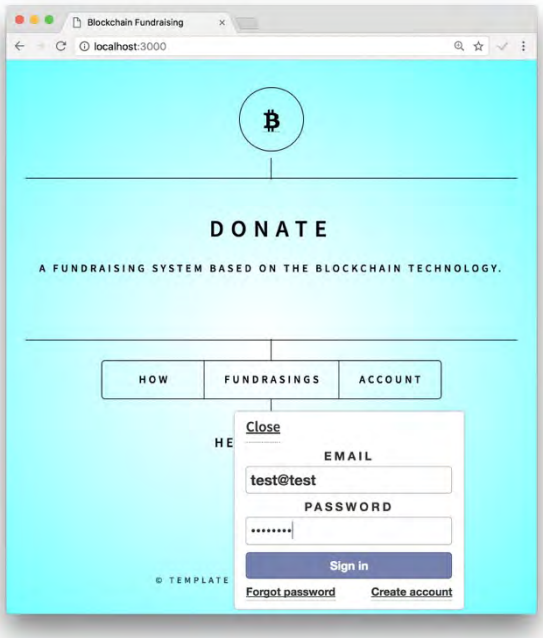
(一) 公開化所有捐款用途

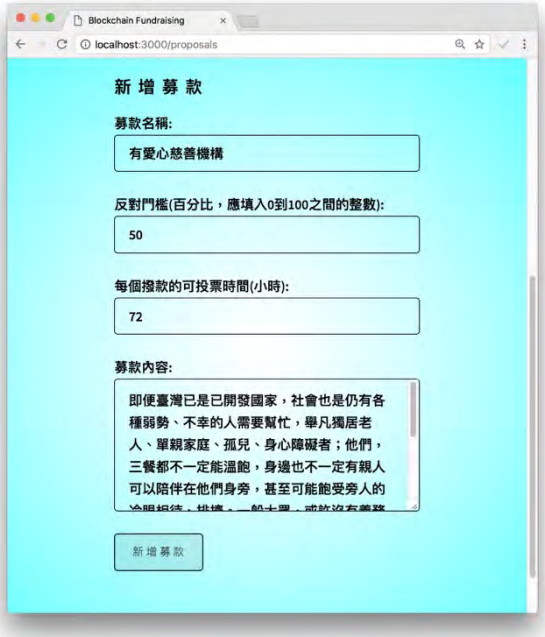
(二) 可監督用款的投票機制

二、利用 Solidity 語言設計出符合上述募捐平台特性的 Smart Contract。

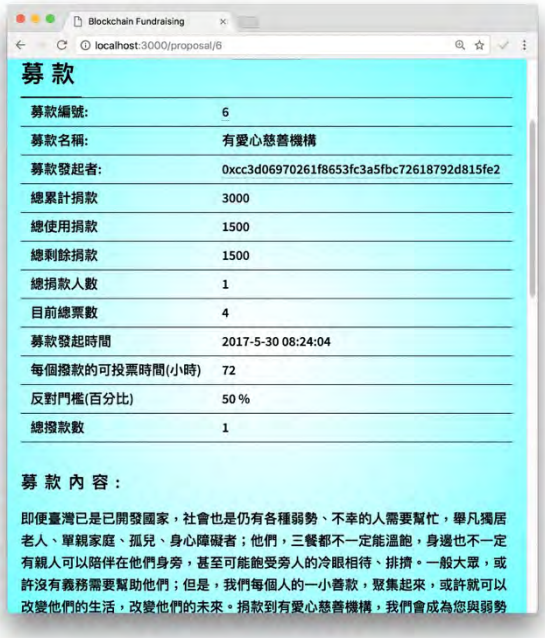
三、成功將平台的 Smart Contract 部署在 Ethereum 上，確認了系統的可實行性。

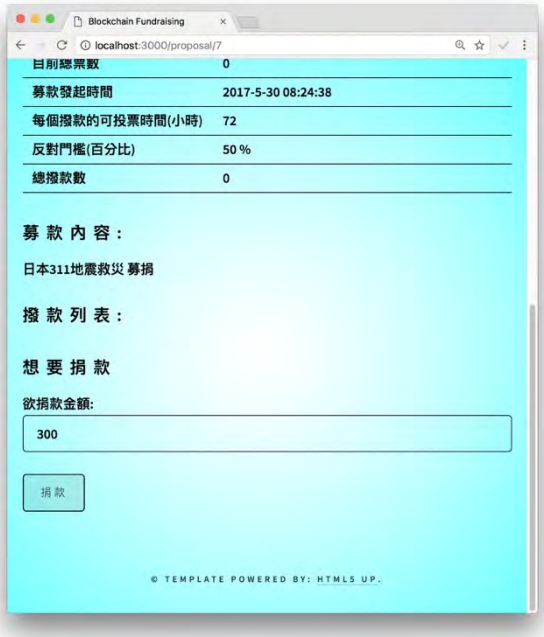
四、利用 Meteor.js 框架與 HTML5-UP 套版與 web3 API 設計出簡易的使用者介面進行以下操作。

	
首頁	可登入或註冊帳號



檢視個人帳戶情形、代幣與法幣交換功能 (還未與第三方合作，兌幣功能暫為模擬的)	募款者可以發起募款





可檢視募款內容	捐款者可捐款至募款

募款的發起者可對該募款進行撥款	可檢視撥款， 募款的捐款者可投票反對該募款的撥款

可檢視所有發起的募款	若撥款成功可將撥款金額撥給受助者

五、综合上述四點，開發出的本平台。可對社會帶來以下貢獻：

(一) 提升慈善機構之責信公信力，改善臺灣公益風氣

(二) 落實捐款者監督，將用款決定權交給捐款者，降低民眾行善之擔憂

(三) 需要募款的個人、非營利組織亦可使用本平台來募款，不僅限於慈善機構。

陸、討論

***Q*:監督機制的投票該採用同意票或是反對票？** *A*:若是使用同意票，每一個撥款都需要捐款人投票通過，若是捐款人投票意願不高，則會造成捐款應用難以施行。故我們最後討論出用反對票來表決。每個捐款者可以投反對票，若不投票則代表同意撥款。
***Q*:為什使用這種計算票數的方式？** *A*:如果捐款與獲得的票數等比例，會造成捐很多錢的人權力過大，但若每個人 1 票又有失公平性。故選擇對捐款金額以 10 為底取對數為票數的方式，盡力使上述兩種狀況平衡。
***Q*:募款發起者的自由度？是否要讓發起者自行決定通過反對門檻以及可投票時間？** *A*:平台上會公告該募款所訂定的規則，可使捐款者自行評估。若是投票反對門檻過高，可預想較少的人願意捐給該募款，而可投票時間過短亦是如此。
***Q*:何時要將代幣從募款中送出？** *A*:發起撥款時就該先將代幣凍結，暫時無法動用那些代幣，需要等投票通過該撥款才會發送至受助者，若反對成功則將款項流回募款；若無如此設計，同一時間可能同時有多筆撥款進行中，在撥款發起時無法確認該募款是否真的有足夠的代幣支付，可能會造成撥款通過後，而募款剩餘款項不足的狀況。
***Q*:募款 A 撥款給募款 B，而誰該擁有這筆款項在募款 B 的投票權？** *A*:因為募款撥款給募款的行為，等同於個人帳戶捐款給募款，只是錢的來源不同。而這筆捐款對於募款 B，**必定需要有人擁有投票權**，否則可能導致募款 B 有捐款卻無人擁有投票權來監督。而最後我們討論出：將投票權給與募款 A 的發起者；若募款 A 的捐款者不滿意將投票權給與募款 A 的發起者，則一樣可行使投票權將該撥款(撥款給募款 B)否決。

Q:本研究與其他 BlockChain 募捐平台差別是？

A:本研究相對於其他 BlockChain 募捐平台，最大的不同是引入了**投票監督機制**，藉由反對票的行使，讓捐款者可以審核募款方每筆款項的使用。

Q:本研究與現今一般募款方式的具體差異？又是使用何種貨幣？

A: 下表分析了一般募款方式與使用 BlockChain 的差異。貨幣方面則有 Smart Contract 自定義代幣與有價值之以太幣兩種選擇。而比較後，自定義貨幣是較為可行的。

	現今一般募款方式	BlockChain（利用 Smart Contract 自定義代幣）	BlockChain（利用有價值之以太幣）
公正第三方兌幣、認證需求	不需要	需要	不需要
一人一帳戶是否為可行的	無意義	可行	**目前不可行**
捐款用途	暗箱	公開透明	公開透明
捐款人能否監督用款	不可以	可以	可以
社會普及與接受程度	成熟，但有疑慮	發展中	發展中
若引入投票監督機制，可否真正信任	不可	可	可

柒、結論

本研究總共利用了 BlockChain、Ethereum、Smart Contract、Meteor.js 框架（與 web3 API）等科技，開發出可監督捐款的公開募捐平台。

BlockChain 的去中心化使捐款用途是完全公開的，而其難以被竄改的特性更可避免資料被更動；而利用了可自行設計的 Smart Contract，設計出一套投票系統來監督募款方每筆用款；且利用 Meteor.js 框架設計出簡易的使用者介面。

利用上述這些技術開發出的募捐平台系統，提高了用款的透明度、降低監督用款的難度，更進一步促進公益責信的成長。然而因為現實上還有各種難關（例如尋找合作機構、法律問題等），未來希望能克服這些困難，使該平台真正貢獻於社會。

捌、未來展望

一、將現實的金流轉換為 Smart Contract 上之交易資訊流。

二、與政府機構或可信任之第三方機構合作，進行使用者現實身分之認證，達到一人一帳號，和代幣與法幣間的交換。

三、與更多合作廠商進行合作，設法將金流之公開部份盡量拉長至實體物件。

四、於校內推廣，進行系統之試營運，蒐集數據與建議，更加完善本系統；之後再更進一步推廣至地方，最後應用於整個社會。

五、考慮更多實務層面的細節問題，更加完善系統完整性。

玖、參考資料

[1]李赫(2016)。<<區塊鏈開發（二）部署和運行第一個以太坊智能合約>>。

取自：http://blog.csdn.net/sportshark/article/details/52249607

[2]EthFans(2015)。<<以太坊智能合約開發之新手教程>>。

取自：http://ethfans.org/posts/101-noob-intro

[3]Andreas M. Antonopoulos(2014). Mastering Bitcoin.

Retrieved from: http://chimera.labs.oreilly.com/books/1234000001802

[4]S Nakamoto(2008).Bitcoin: A Peer-to-Peer Electronic Cash System

Retrieved from: https://bitcoin.org/bitcoin.pdf

[5]V Buterin(2014).Ethereum: A next-generation smart contract and decentralized application platform.

Retrieved from: https://www.weusecoins.com/assets/pdf/library/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[6]Solidity Documentation

Retrieved from: https://media.readthedocs.org/pdf/solidity/develop/solidity.pdf

【評語】052505

本作品利用區塊鍊的技術來實現可監督捐款的公開募捐平台，完整性高、具實用價值。

但目前作品只是利用相關開發工具進行應用的功能開發，欠缺實驗數據的分析和討論。

建議進行多方面的科學討論和實驗來探討此作品的效能，例如當參與人數越來越多時，投票和否決所需的時間會如何變化。

作品海報

壹、研究動機

臺灣公益責信協會於 2015 年統計，全臺六百大非營利組織基金會，有高達八成未提供財務報告，尤其許多慈善機構不常公開捐款用途；近年也有新聞報導指出，有些慈善機構將民眾善款用於非公益事項；層出不窮的事件引起我們反思：難道捐款者沒有辦法知道自己的善心究竟被用到哪了嗎？莫非捐款者沒有監督捐款用途的權利嗎？

貳、研究目的

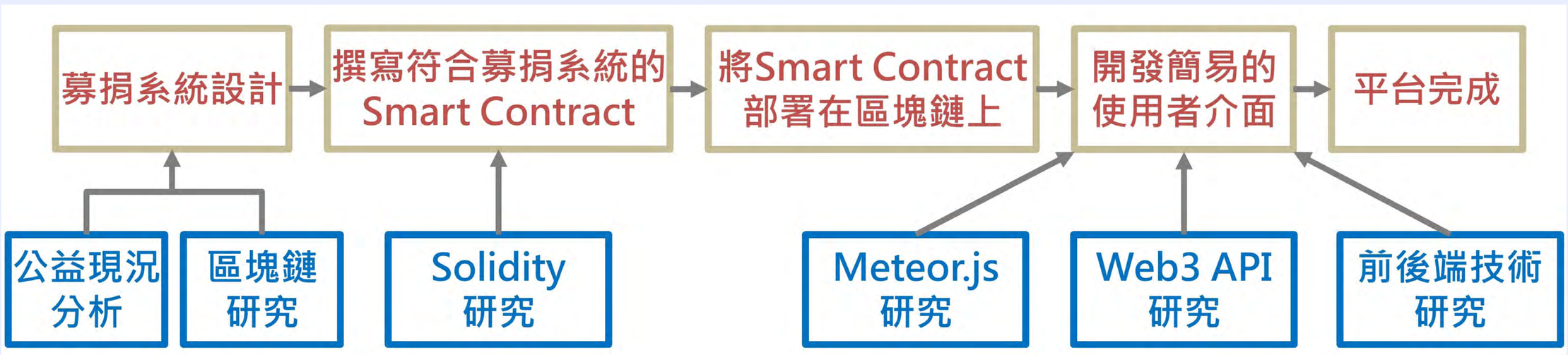
一、公開且無法竄改：利用 BlockChain 使**捐款用途公開化**且**資料無法被竄改**
二、用戶的監督權利：藉由 Ethereum Smart Contract 達成令捐款者**擁有投票權監督所有用款**
三、簡便的操作介面：研究網頁前後端程式語言，設計出**簡易的 UI**(使用者介面)
四、改善責信公信力：**提升公益責信度**，讓社會大眾能夠更放心得捐款

參、研究設備及器材

硬體設備：紙、筆、電腦、伺服器。 **軟體：** Remix、Ethereum Mist、Sublime Text。

肆、研究過程或方法

• 募捐平台研究步驟圖



• 募捐系統概念設計

1. 用戶：平台中有一般用戶與管理員兩種，管理員具有公正交換法幣與代幣之權限。
2. 募款：而每一個用戶皆可進行依其目的或機構名稱進行募款，例如**「xx 慈善機構」即為一種募款**，「幫助地震災民」也是一種。在募款時應提供發起撥款時投票反對門檻與可投票時間供用戶決定是否捐款。而每個募款應要填寫:

   • 募款名稱 • 募款內容 • 反對門檻 • 募款中每個撥款的可投票時間(小時)

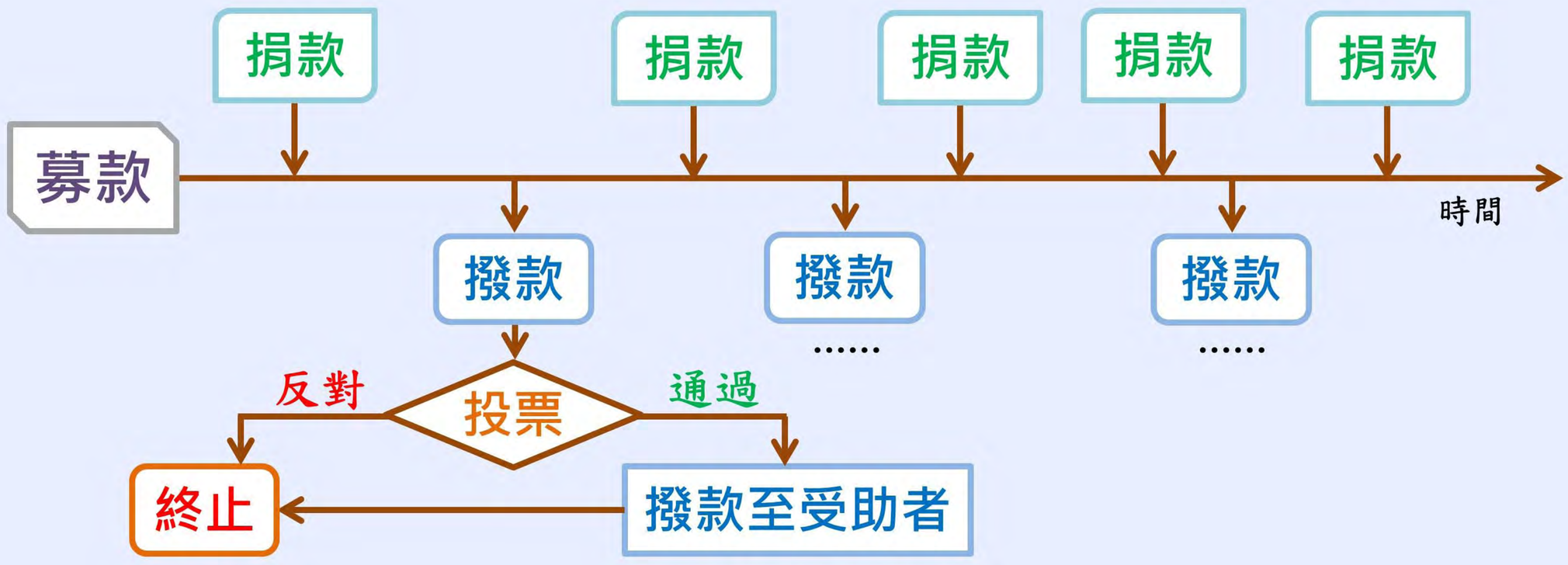
   ※反對門檻與可投票時間須且僅能在募款發起時就設定好
3. 捐款：每一個募款，所有用戶都可以捐款，有捐款至該募款的用戶，便具有監督該募款使用款項的權利。
4. 撥款：每一個募款發起者皆可以在其募款中依其募款所募得之善款進行多次撥款。而每個撥款應要填寫:

   • 撥款名稱 • 撥款內容 • 撥款金額 • 受助者
5. 投票監督：在發起該撥款後，該時刻前所有有向該募款捐款的人，在可投票時間內都有向該撥款投反對票的權利。若某個用戶在該募款中的捐款金額為正整數 D，則該捐款金額可獲得之票數為

$$floor(\ log10\ (D)\ )+1$$

即 D 以 10 為底取對數向下取整再加 1，而一人想獲得絕大多的票是極難的。若反對總票數的在總票數的百分比超過了反對門檻則不會順利撥款，否則會順利撥款至受助者。

• 募捐平台設計架構圖



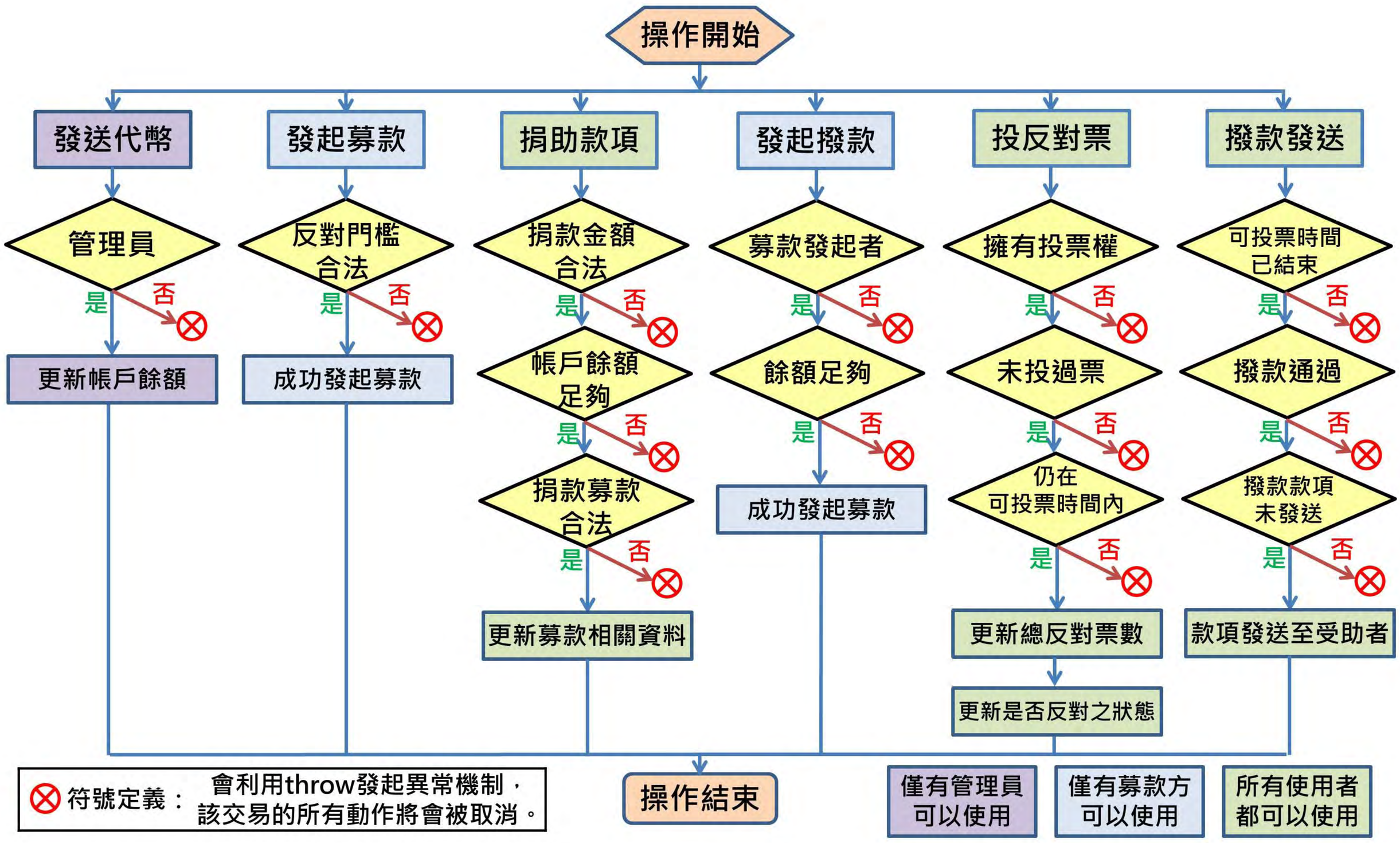
• 與實務串接

若要利用法幣做捐款，而現行的法幣都還不是數位貨幣的形式，導致 BlockChain 上記載的代幣僅是代表法幣的資訊，故必須有個可信任的第三方機構來做代幣與法幣 1:1 的交換。且須要做身份認證，達到一人一帳戶而管理員為可信任的第三方機構，例如:銀行、政府機關等等。

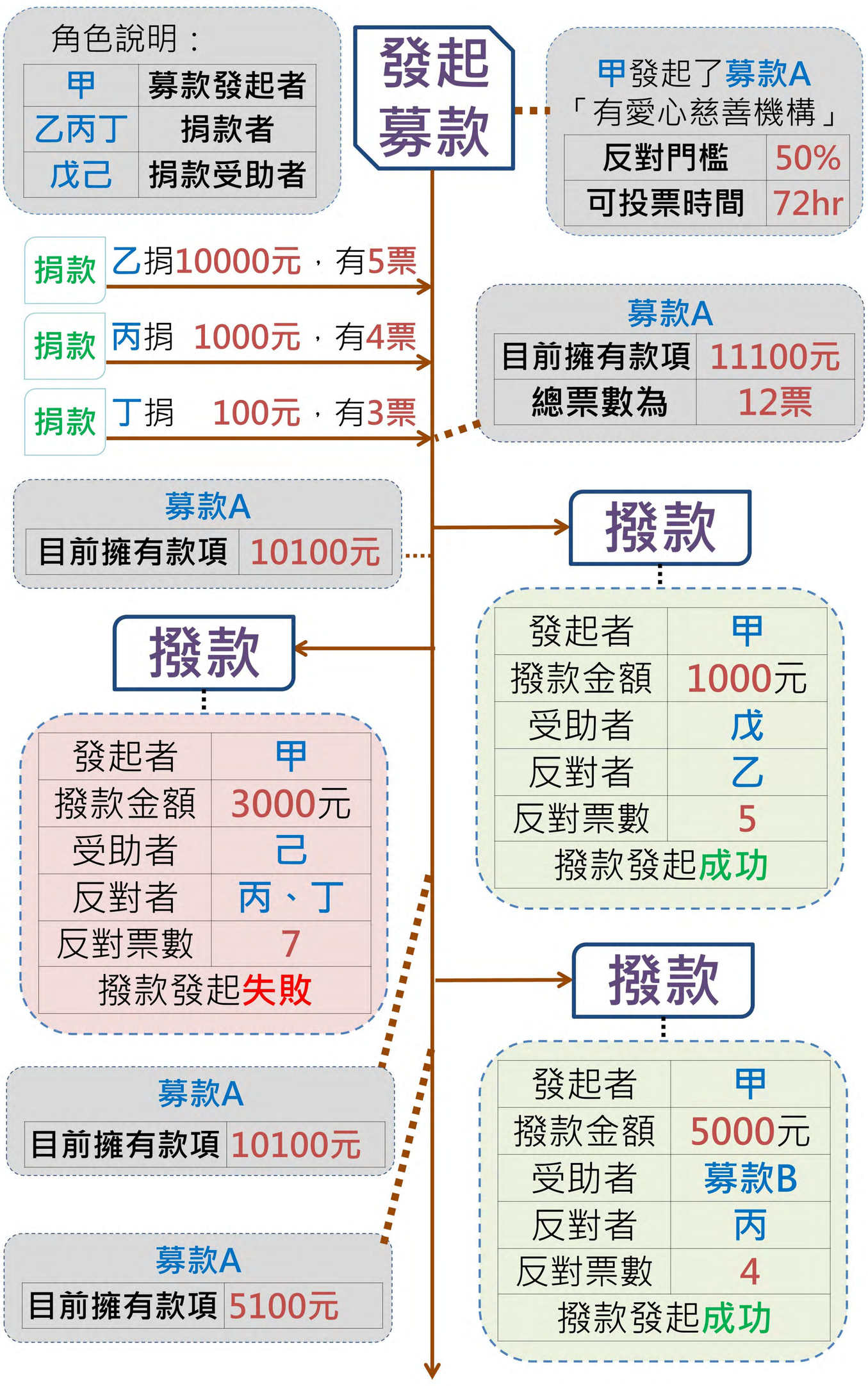
• Smart Contract

為了使整個平台的運行規則不可竄改，就像是定好一個合約，放在區塊鏈上，使所有執行的操作都是可信任的。

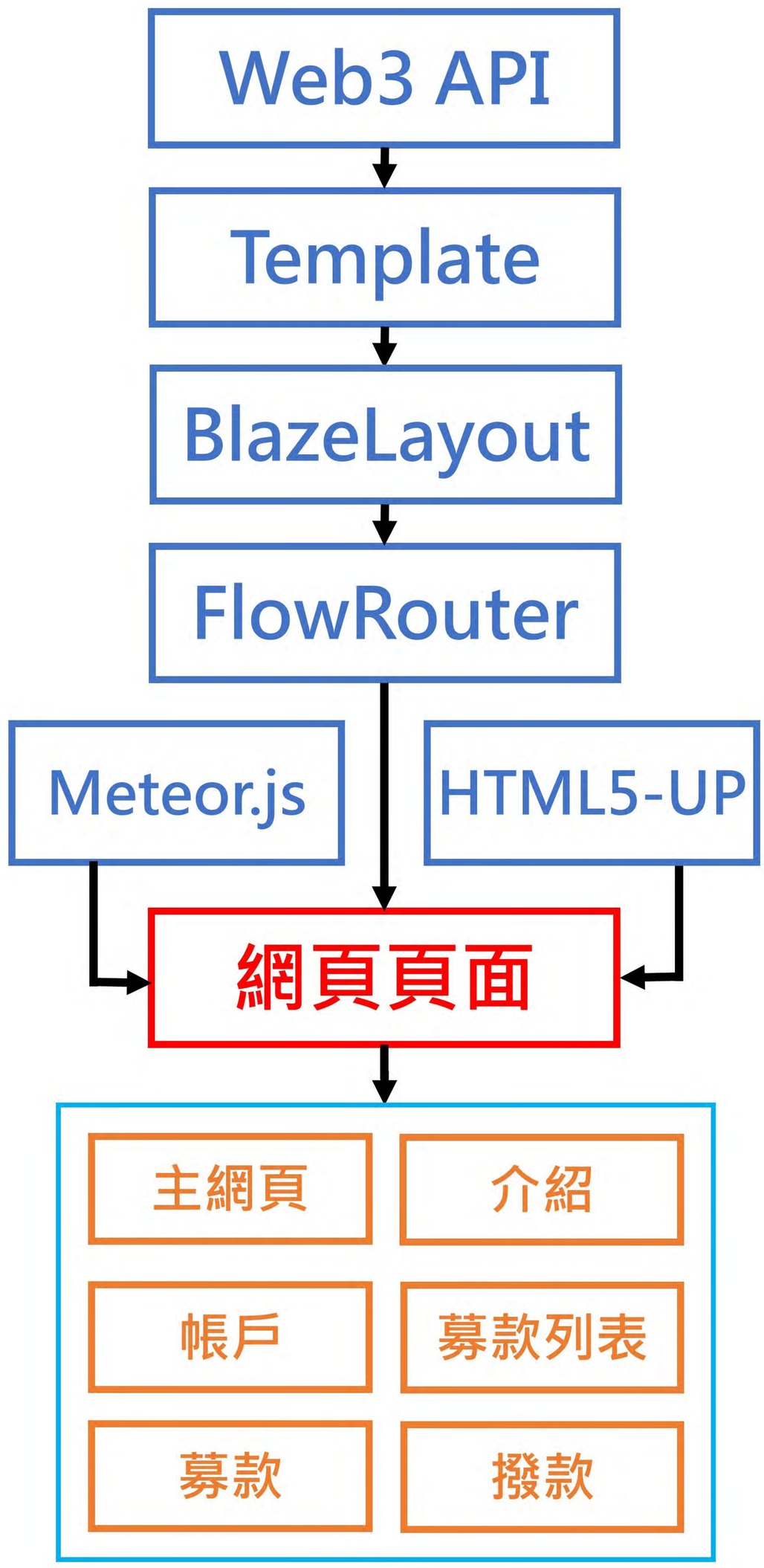
• Smart Contract 流程圖



• 募捐實例圖



• UI 架構圖



伍、研究結果

一、藉由 Smart Contract 與 Ethereum BlockChain ，設計了一套有以下特點的募捐平台。

(一) **公開化所有捐款用途**

(二) **可監督用款的投票機制**

二、利用 Solidity 語言設計出符合上述募捐平台特性的 Smart Contract。

三、成功將平台的 Smart Contract 部署在 Ethereum 上，確認了系統的可實行性。

四、利用 Meteor.js 框架與 HTML5-UP 套版與 web3 API 設計出簡易的使用者介面以使用本平台。

五、綜合上述四點，開發出的本平台。可對社會帶來以下貢獻：

(一) **提升慈善機構之責信公信力**，改善臺灣公益風氣

(二) 落實捐款者監督，將用款決定權交給捐款者，降低民眾行善之擔憂

(三) 需要募款的個人、非營利組織亦可使用本平台來募款，不僅限於慈善機構。

陸、討論

• 三種募捐實現方法比較

	現今一般募款方式	BlockChain (利用 Smart Contract 自定義代幣)	BlockChain (利用有價值之以太幣)
公正第三方兌幣、認證需求	不需要	需要	不需要
一人一帳戶是否為可行的	無意義	可行	**目前不可行**
捐款用途	暗箱	公開透明	公開透明
捐款人能否監督用款	不可以	可以	可以
社會普及與接受程度	成熟，但有疑慮	發展中	發展中
若引入投票監督機制，可否真正信任	不可	可	可

柒、結論

本研究總共利用了 **BlockChain**、**Ethereum**、**Smart Contract**、**Meteor.js 框架**(與 **web3 API**)等科技，開發出可監督捐款的公開募捐平台。

BlockChain 的去中心化使捐款用途是完全公開的，而其難以被竄改的特性更可避免資料被更動；而利用了可自行設計的 Smart Contract，設計出一套投票系統來監督募款方每筆用款；且利用 Meteor.js 框架設計出簡易的使用者介面。

利用上述這些技術開發出的募捐平台系統，提高了用款的透明度、降低監督用款的難度，更進一步促進公益責信的成長。然而因為現實上還有各種難關(例如尋找合作機構、法律問題等)，未來希望能克服這些困難，使該平台真正貢獻於社會。

捌、未來展望

一、將現實的金流轉換為 Smart Contract 上之交易資訊流。

二、與政府機構或可信任之第三方機構合作，進行使用者現實身分之認證，達到**一人一帳號**，和代幣與法幣間的交換。

三、與更多合作廠商進行合作，設法將金流之公開部份盡量拉長至實體物件。

四、於校內推廣，進行系統之試營運，蒐集數據與建議，更加完善本系統;之後再更進一步推廣至地方，最後應用於整個社會。

五、考慮更多實務層面的細節問題，更加完善系統完整性。

玖、參考資料

[1]李赫(2016)。<<區塊鏈開發（二）部署和運行第一個以太坊智能合約>>。取自:http://blog.csdn.net/sportshark/article/details/52249607

[2]EthFans(2015)。<<以太坊智能合約開發之新手教程>>。取自:http://ethfans.org/posts/101-noob-intro

[3]Andreas M. Antonopoulos(2014). Mastering Bitcoin. Retrieved from: http://chimera.labs.oreilly.com/books/1234000001802

[4]S Nakamoto(2008).Bitcoin: A Peer-to-Peer Electronic Cash System. Retrieved from: https://bitcoin.org/bitcoin.pdf

[5]V Buterin(2014).Ethereum: A next-generation smart contract and decentralized application platform. Retrieved from: https://www.weusecoins.com/assets/pdf/library/Ethereum_white_papera_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[6]Solidity Documentation . Retrieved from: https://media.readthedocs.org/pdf/solidity/develop/solidity.pdf