110 學年 第 2 學期 金融科技安全 期末專題報告 日期:2022/6/23

(專題組員為 1 至 2 人為限)

班級:	**資工三**	**學號:**	**B0843026**	**姓名:**	**謝奇容**
班級:	**資工三**	**學號:**	**B0943202**	**姓名:**	**游于函**

1. 區塊鍊智能合約應用程式開發 (共 1 題，100 分，滿分 100 分)

請以課程中所教授智能合約範例為基礎，構思一 Ethereum 智能合約應用，並透過 Remix IDE + Metamask 部署於 Test Network。

請於 2022/6/23 23:59 前透過數位學習園區繳交報告及程式碼

A. 描述本專題的題目、動機、目的與架構。

題目：基於智能合約之就學貸款應用

動機：

政府於民國六十五年開辦助學貸款，民國八十三年時更名為就學貸款，其目的在於協助中低收入戶的家庭子女入學，減輕其教育的經費負擔，也促使教育機會平等。因此，在就讀大學期間，有許多同學會向銀行申請就學貸款，來減輕家裡的經濟壓力也讓自己可以順利讀完大學，等到畢業工作後再來還這些貸款。雖然申請就學貸款對於家裡經濟較不好的同學很友善，讓每個人都有機會讀大學，但是因為申請學貸要每個學期都親自到銀行申請，是有點麻煩的，而且在疫情期間我們應該要減少外出活動，降低染疫的風險。

現在虛擬加密貨幣及智能合約的應用越來越普遍，透過其不可串改及透明的特性，可以讓學生清楚的看到貸款的金額和明細等資料，如此一來就能夠確保自己的貸款資訊。因此，我們將實作出一個利用智能合約來讓學生不用出門就可以申請學貸的機制，也讓銀行、學校、學生三方清楚的知道金錢的流向以及貸款的金額。

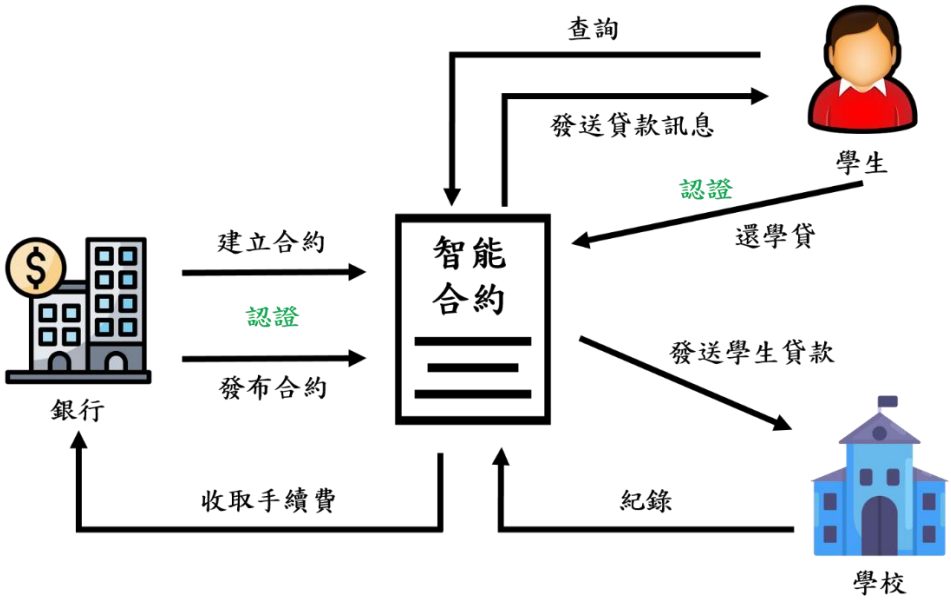
目的：

本專題的目的是希望利用智能合約的方式改善學生每學期都要親自到銀行申請學貸的不便利，我們透過區塊鏈記錄每筆的交易紀錄，利用區塊鏈上資料難以竄改的特性，確保資料的完整性與準確性，並且就學貸款是銀行和學生所簽訂的，具有唯一性，而智能合約也具有唯一性的性質，因此符合就學貸款所需的條件，且這些資料都是可以被銀行、學校和學生所存取，達到真正的透明公開，確保學生貸款的內容以及之後還款的金額。

架構：

圖一為此智能合約的架構圖。總共會有四個角色來進行互動，分別為智能合約、銀行、學校以及學生，智能合約是由銀行來建立以及發布，發布的同時智能合約會向銀行收取手續費，接著銀行會將學生的資料寫入合約中，內容包含學生位址、學校位址、學生身分證字號、學校名稱等基本資料，註冊完成後，銀行每學期會發送學生貸款給學校，同時新增一筆學生的貸款資訊，內容包含學生位址、

學生身份證字號、貸款金額、手續費和貸款時間，而貸款金額和手續費會存入學生資料中的總貸款金額，當學生畢業繳款時會自動將貸款總金額扣除，讓學生知道他還有多少貸款要還。

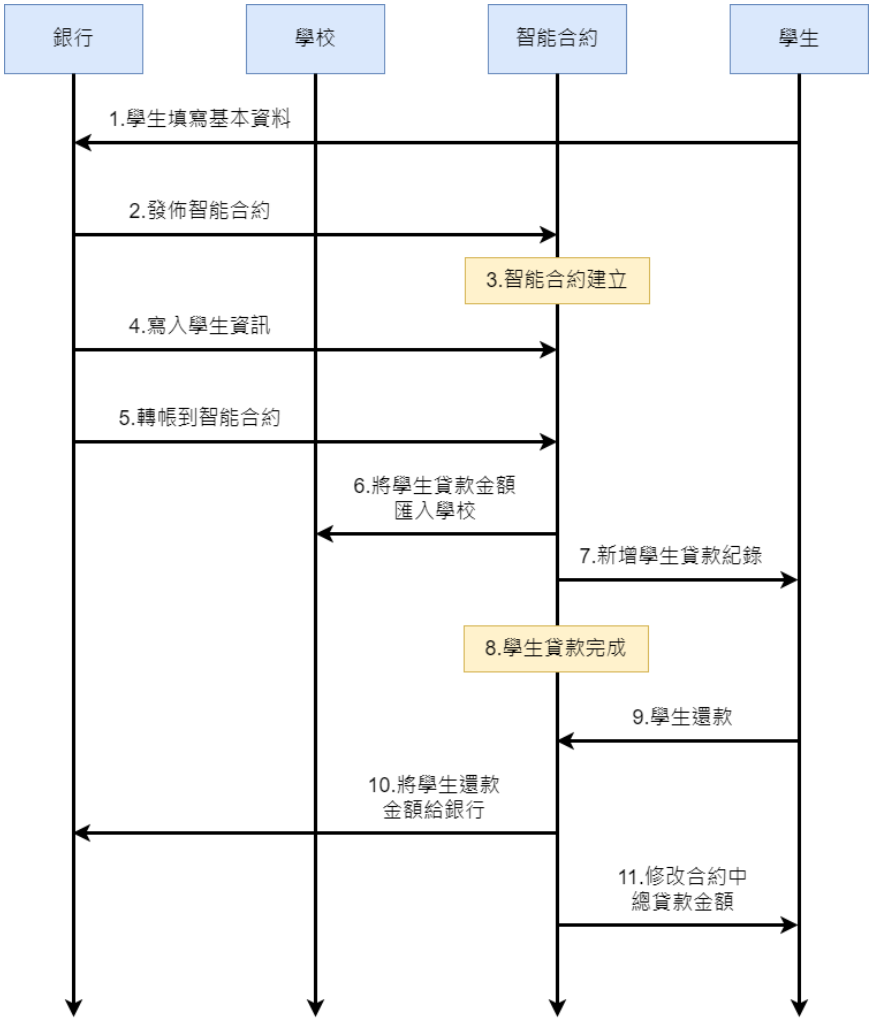


圖一、合約架構圖

圖二是此智能合約的序列圖。如果學生要申請就學貸款必須先建立個人的乙太幣錢包也就是帳號，而銀行則需要發佈智能合約讓學生可以完成貸款的作業流程。

每個學期智能合約會將學生的貸款金額給學校，並且在學生的紀錄上記上一筆貸款的金額，學生也可以從上面看到他所貸款的資料，以及總貸款金額，這樣學生的就學貸款就完成了。

當學生要開始還貸款時，會先把錢給智能合約，合約再將錢給銀行並且修改合約中學生的總貸款金額，這個過程會在學生還款時執行，直到學生把整個貸款金額都還完。



圖二、合約序列圖

B. 解說本專題程式的開發環境、關鍵程式碼。

開發環境：

本專題使用 solidity 撰寫智能合約，利用 remix IDE 作為開發環境，並將合約部屬在 Ropsten 的測試網路上。

關鍵程式碼：

首先是合約中的變數與限制的介紹，包含建構子與資料結構。一開始會先定義使用的環境，此合約可使用 0.4.22 ~ 0.9.0 的 solidity 環境執行。接著是建構 loan 這個學貸的合約，合約內先定義會一直保存在合約內的三個變數，銀行帳戶、銀行名稱、學生數量，如圖三所示。

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity >= 0.4.22 < 0.9.0;
3
4  contract loan {
5      address public bank;
6      string public bankName;
7      uint private studentNum;
```

圖三、環境宣告與全域變數設置

接著，我們建立兩個資料結構，student 存放學生的個人資訊，studentLoan 存放學生每次貸款的明細。而最下方的三個陣列，第一個 students 是存放所有學生的資料，第二個 sL 是存放每一位學生的總貸款資料，第三個 allLoanDetails 是存放所有學生的貸款明細，如圖四所示。studentLoan 中的 interest 與 done 兩個變數是未來計算利息時使用，本次實驗預設學生皆為中低收入戶，不需繳交利息。

```
9   struct studentLoan{
10      address studentAddress;
11      string studentID;
12      uint tuiion;
13      uint handlingFee;
14      uint startTime;
15      uint interest;      //此筆貸款是否收取利息，0為不收，數字為利息比率（因會照學生家庭收入收取不同利息）
16      uint done;          //是否還完此筆貸款，0未繳清，1繳清
17  }
18
19  struct student{
20      address studentAddress;
21      address schoolAddress;
22      string studentID;
23      string schoolName;
24      string studentName;
25      uint allLoanMoney;
26      uint loanNum;
27  }
28
29  student[] students;
30  studentLoan[] sL;
31  studentLoan[][] allLoanDetails;
```

圖四、資料結構與陣列

再來是智能合約的建構子，此合約由銀行發佈，發佈合約時需使用銀行的錢包，並填入銀行的名稱，最後設定此時貸款學生人數為 0，如圖五所示。

```
33      constructor (string memory _bankName)  {    //建構子
34          bank = msg.sender;
35          bankName = _bankName;
36          studentNum = 0;
37      }
```

圖五、智能合約建構子

為了避免有惡意人士或不良學生任意更改貸款資料，我們有作身分判別，如圖六所示。註冊學生資料、新增貸款資料與存款，只有銀行可以使用，非銀行帳號使用時，交易會失敗，資料也不會更動。查詢個人資料、查詢剩餘金額、還款與更改錢包位址，只有學生本人可以使用，非學生本人使用，交易會失敗，資料也不會更動。

```
39      modifier onlyBank{
40          require(msg.sender == bank);
41          _;
42      }
43
44      modifier onlyStudent{
45          bool qual = false;
46          for (uint i=0; i<studentNum; i++){
47              if(students[i].studentAddress == msg.sender){
48                  qual = true;
49              }
50          }
51
52          require(qual == true);
53          _;
54      }
```

圖六、身分判別函式

再來是銀行可以操作的三個功能，分別是存款、註冊學生資料與新增貸款資料。合約發佈後，銀行需存入足量金額，讓新增貸款成功後，合約內有足夠金額能轉帳給學校。

```
73      function saveMoney()public payable onlyBank returns(uint) {
74          return msg.value;
75      }
```

圖七、存款

註冊學生資料時，需填入學生錢包位址、就讀學校錢包位址、學生身份證字號、學校名稱、學生姓名和目前的貸款金額，如圖八所示。此時合約會於 students 陣列中新增一筆學生資料，包含上述輸入的所有資訊，並填上目前貸款資料為 0 筆。allLoanDetails 中也會新增一個貸款資料的空陣列，等新增貸款資料後填入貸款資料。最後要在學生數量的計數器 studentNum 上+1。

```
56  function register(address _studentAddress, address _schoolAddress, string memory _studentID, string memory _schoolName,
57                                                  string memory _studentName, uint _amount) public onlyBank{  //註冊學生資料
58      students.push(student({
59          studentAddress : _studentAddress,
60          schoolAddress : _schoolAddress,
61          studentID : _studentID,
62          schoolName : _schoolName,
63          studentName : _studentName,
64          allLoanMoney : _amount,
65          loanNum : 0
66
67
68      }));
69      allLoanDetails.push(sL);
70      studentNum ++;
71  }
```

圖八、註冊學生資料

銀行的第二個功能是新增學生貸款資料，新增資料時需填入學生錢包位址、學生身份證字號、學費金額、手續費金額、學期開始時間、是否收取利息，如圖九所示。利用學生錢包位址，找到該名學生後，在他的貸款資料中新增一筆資料，填入輸入的資訊，並設為未繳清狀態。資料成功新增後，合約就會將這筆學費轉帳至學生登記的學校錢包位址。我們合約預設單位為 eth，但 solidity 的單位是 wei，故轉帳時需要進行轉換。

```
77      function newLoan(address _studentAddress, string memory _studentID, uint _tuition, uint _handlingFee,
78                               uint _startTime, uint _interest) public onlyBank{  //新增貸款資料
79          for (uint i=0; i < studentNum ; i++){
80              if (students[i].studentAddress == _studentAddress){
81                  students[i].allLoanMoney += _tuition + _handlingFee;
82                  allLoanDetails[i].push(studentLoan({
83                      studentAddress : _studentAddress,
84                      studentID : _studentID,
85                      tuiion : _tuition,
86                      handlingFee : _handlingFee,
87                      startTime : _startTime,
88                      interest : _interest,
89                      done : 0
90                  }));
91                  toSchool(students[i].schoolAddress, _tuition * 1000000000000000000);
92                  break;
93              }
94
95          }
96      }
97
98      function toSchool(address _schoolAddress, uint _tuition)internal onlyBank returns(uint){
99          payable(_schoolAddress).transfer(_tuition);
100         return(_tuition);
101     }
```

圖九、新增貸款資料

最後，是學生可以使用的四個功能，分別是還款、更改錢包位址、查詢剩餘款項與查詢個人資訊。此合約內，尚未定義還款的期限和每期應繳金額，讓學生採自由還款的方式。收到學生款項後，先判斷學生身份，再判斷款項是否大於未繳清金額，若小於未繳清金額，則將這些轉帳給銀行，讓銀行可以做後續的應用，未來也可以設定為留在合約內，供後續貸款交易使用。若還款金額大於未繳清金額，將還款金額轉帳給銀行，剩餘款項還給學生，如圖十所示。

```
102     function repayment(string memory _studentID) public payable onlyStudent returns(string memory name, uint amount){  //還款
103         for (uint i=0; i < studentNum ; i++){
104             if (students[i].studentAddress == msg.sender && keccak256(abi.encodePacked(students[i].studentID)) == keccak256(abi.encodePac
105                 if (msg.value <= (students[i].allLoanMoney) * 1000000000000000000){
106                     students[i].allLoanMoney -= (msg.value / 1000000000000000000);
107                     payable(bank).transfer(msg.value);
108                 }
109                 else if ((students[i].allLoanMoney) * 1000000000000000000 < msg.value) {
110                     uint money = msg.value;
111                     money -= (students[i].allLoanMoney / 1000000000000000000);
112                     payable(bank).transfer(students[i].allLoanMoney * 1000000000000000000);
113                     students[i].allLoanMoney = 0;
114                     payable(msg.sender).transfer(money);
115                 }
116                 return(students[i].studentName, students[i].allLoanMoney);
117             }
118         }
119     }
```

圖十、學生還款

學生可以自行更改錢包位址，需使用原設定錢包提出申請，並填入新錢包的位址，如圖十一所示。更新成功後，舊錢包無法執行任何動作，所有學生功能將只能由新錢包執行。

```
121     function change_address(address _changeAddress) public onlyStudent{  //更改學生錢包位址
122         for(uint i=0; i<studentNum; i++){
123             if(students[i].studentAddress == msg.sender){
124                 students[i].studentAddress = _changeAddress;
125             }
126         }
127     }
```

圖十一、更改錢包位址

學生可以查詢剩餘款項，此功能會從學生資料中查詢，顯示學生錢包位址、學生姓名與剩餘款項，如圖十二所示。若查詢不到此學生，會回傳 Can't Search

的文字訊息，提醒使用者。

```
129     function remainingAmount() public onlyStudent view returns(address, string memory, uint){  //查詢剩餘未繳清金額
130         for(uint i=0; i<studentNum; i++){
131             if(students[i].studentAddress == msg.sender){
132                 return(students[i].studentAddress, students[i].studentName, students[i].allLoanMoney);
133             }
134         }
135         return(msg.sender,"Can't Search",0);
136     }
```

圖十二、查詢剩餘金額

最後，是學生查詢個人資料的功能，此功能會列出較多學生登記的個人資訊，比剩餘款項多了學校名稱和學校位址，讓學生可以確認自己所登記的個人資訊，如圖十三所示。若查詢不到此學生，會回傳 Can't Search 的文字訊息，提醒使用者。

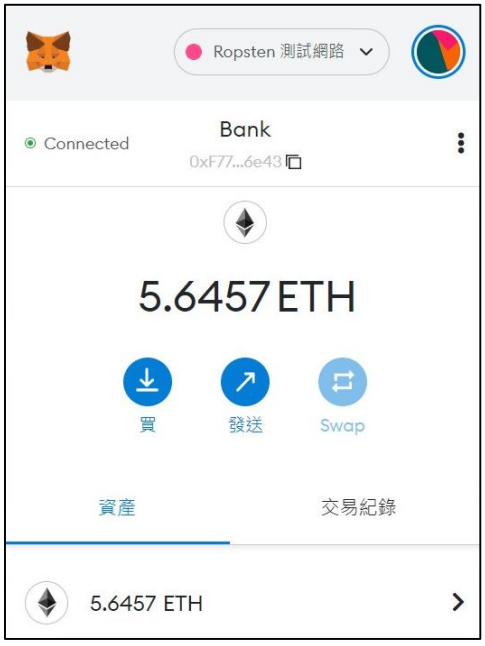
```
138     function personalData() public onlyStudent view returns(address,string memory, string memory, uint, string memory){
139         for(uint i=0; i<studentNum; i++){
140             if(students[i].studentAddress == msg.sender){
141                 return(students[i].studentAddress, students[i].studentName, students[i].schoolName, students[i].allLoanMoney, students[i].studentID);
142             }
143         }
144         return(msg.sender,"Can't Search", "", 0, "");
145     }
```

圖十三、查詢個人資料

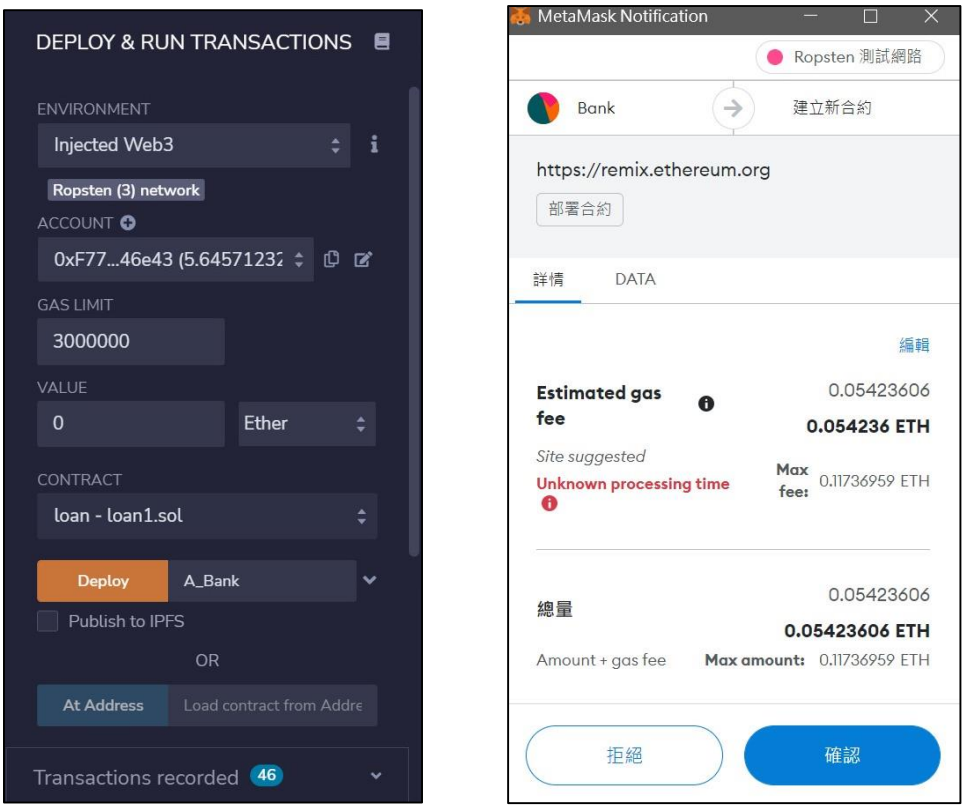
C. 呈現本專題程式的執行結果。

★銀行的部分

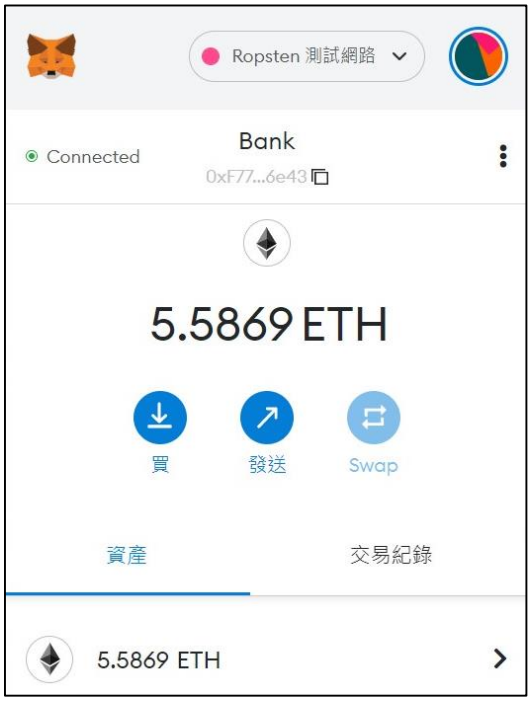
1.銀行填寫完資料後，發佈智能合約。



▲銀行的初始帳戶金額

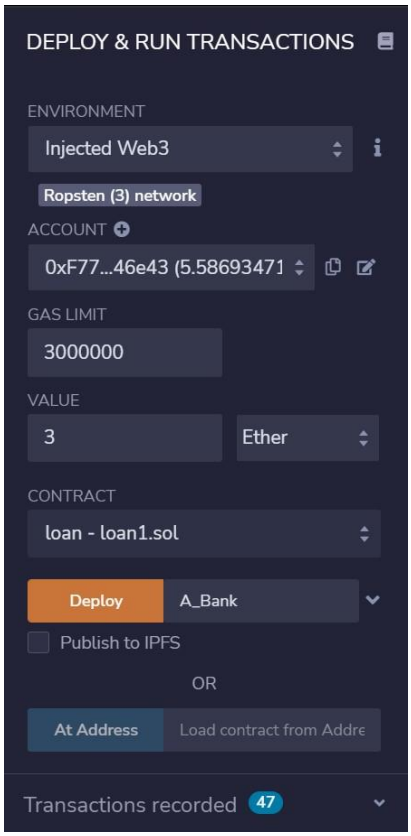


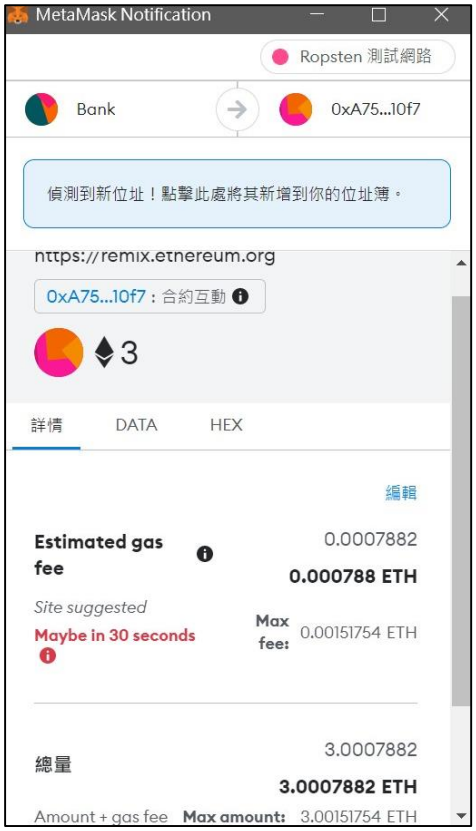
▲銀行填寫資料並發佈合約



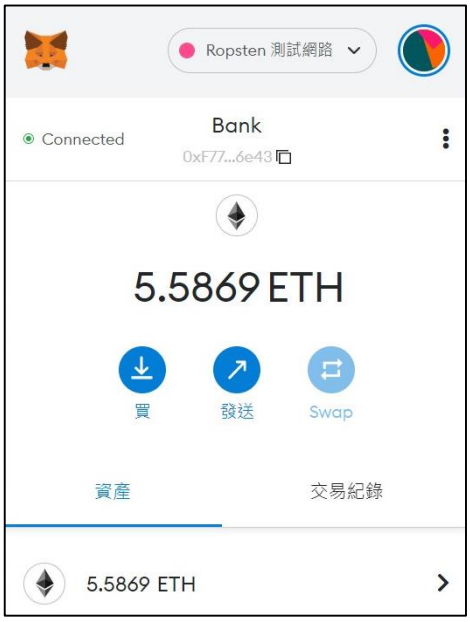
▲合約發佈完成，扣除手續費所剩的錢

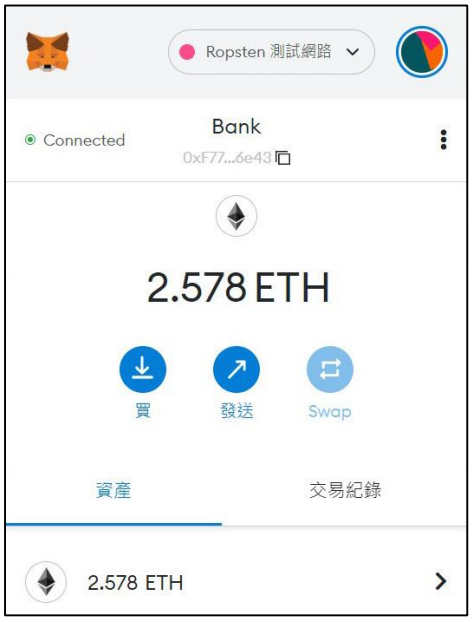
2.銀行存入 3ETH 到智能合約。





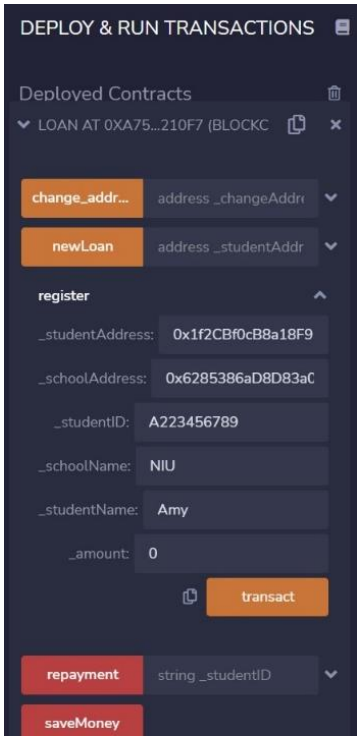
▲銀行存入 3ETH

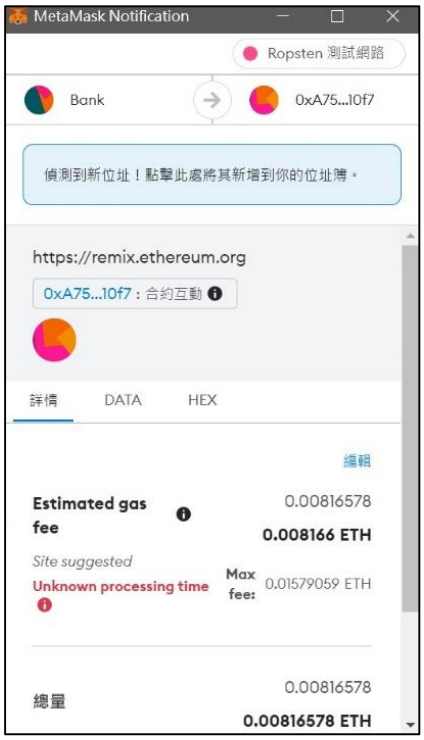




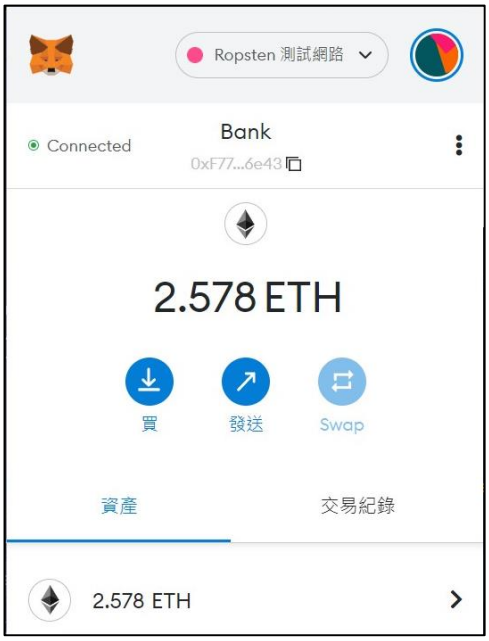
▲存入 3ETH 後，銀行帳戶金額之變化

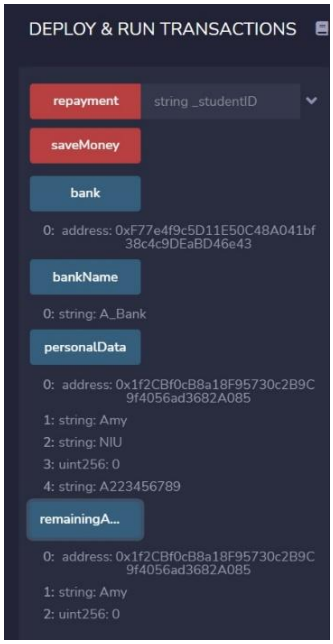
3.註冊學生貸款資料。





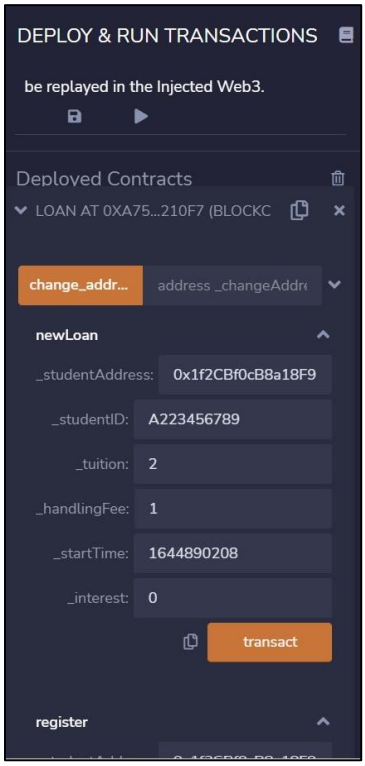
▲輸入學生資料以及所扣的手續費

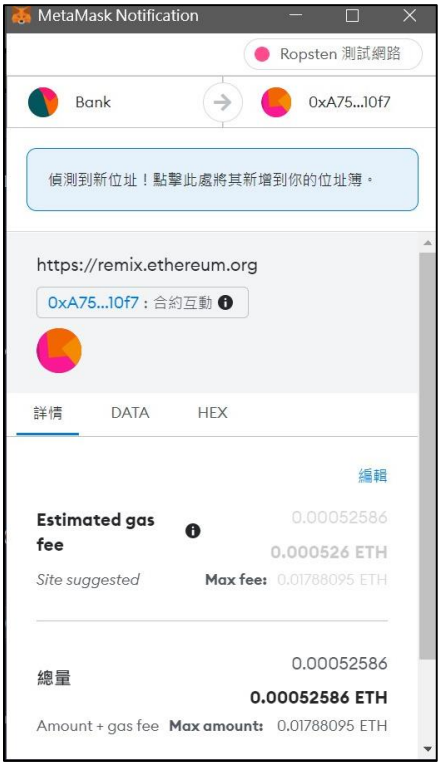




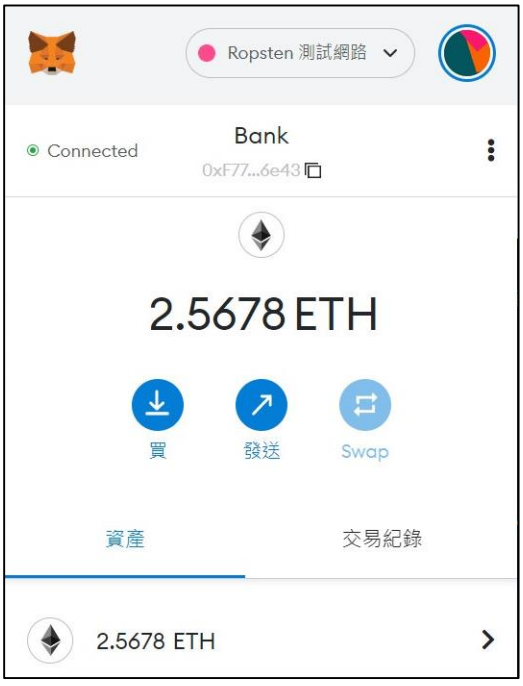
▲成功後銀行所剩的餘額，以及可以查看學生的註冊資料

4.新增學生貸款資料。





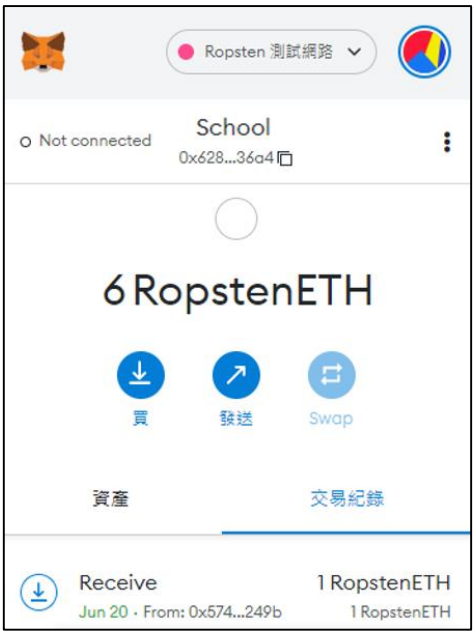
▲新增學生貸款基本資料以及所付的手續費



▲新增成功後銀行所剩金額，以及可以查看學生的貸款資訊

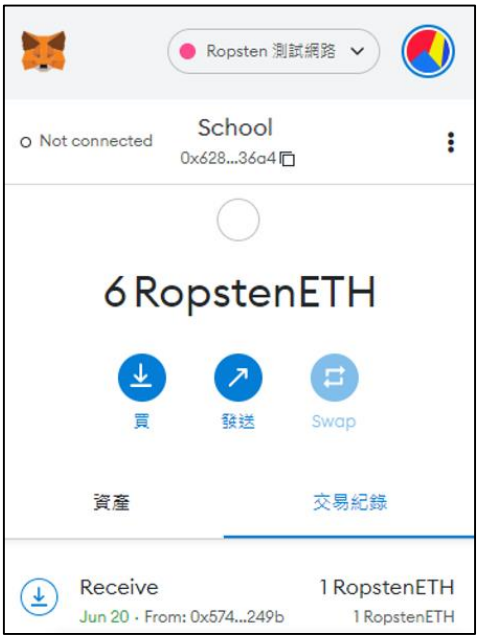
★學校的部分

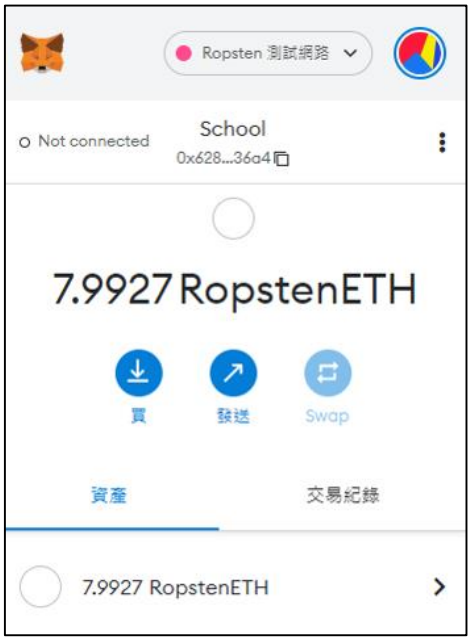
1.學校建立帳戶



▲學校原始金額

2.銀行將學生貸款匯入學校錢包

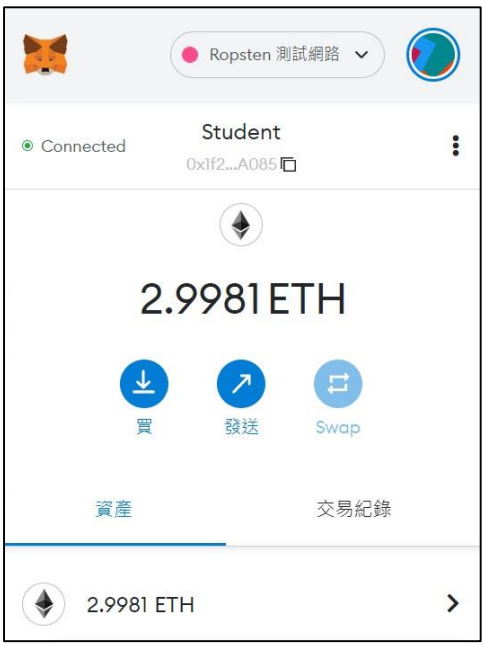




▲學校收到錢後，帳戶金額之變化

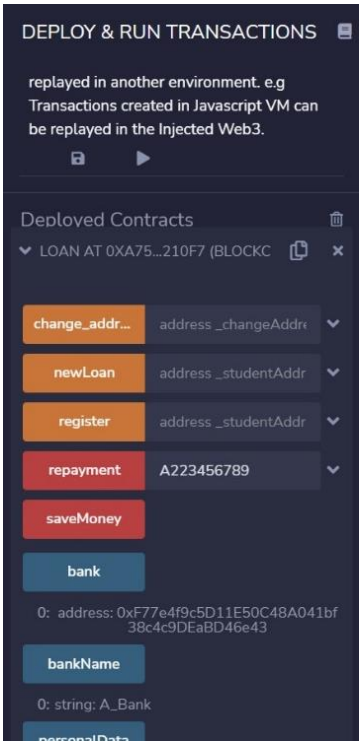
★學生的部分

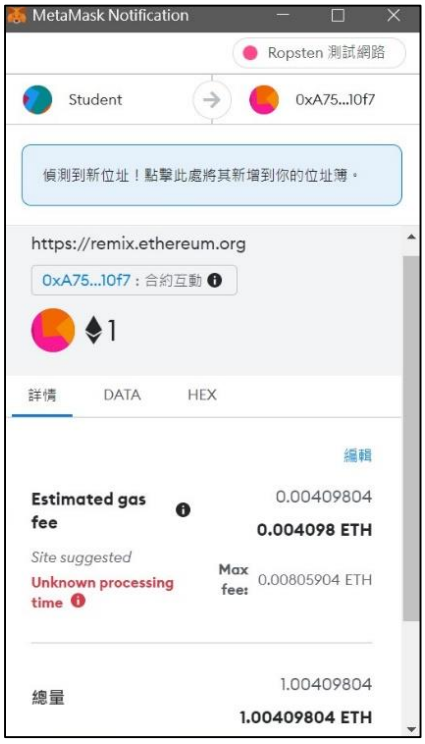
1.學生建立帳戶



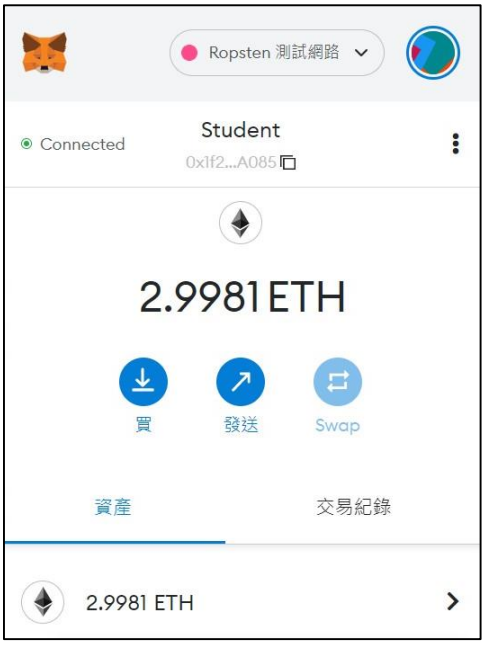
▲學生一開始錢包的金額

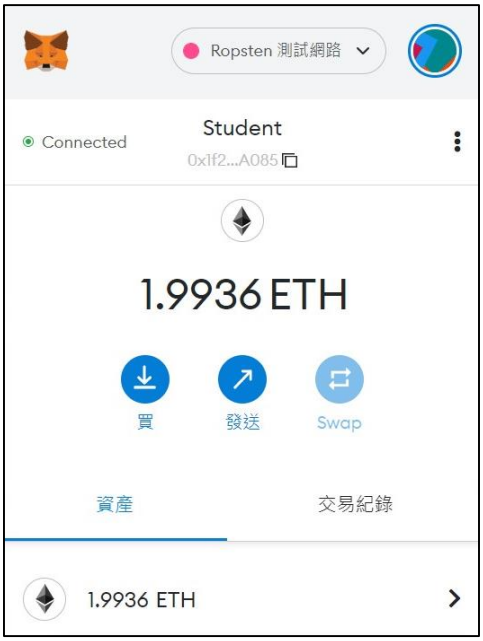
2.學生畢業後，開始償還部分貸款





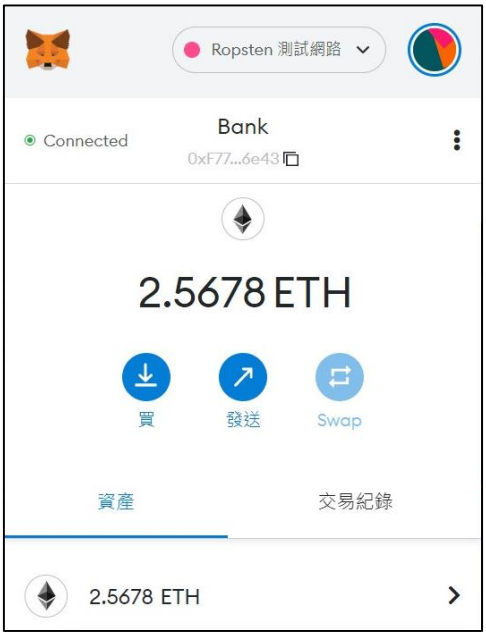
▲學生還款的資料

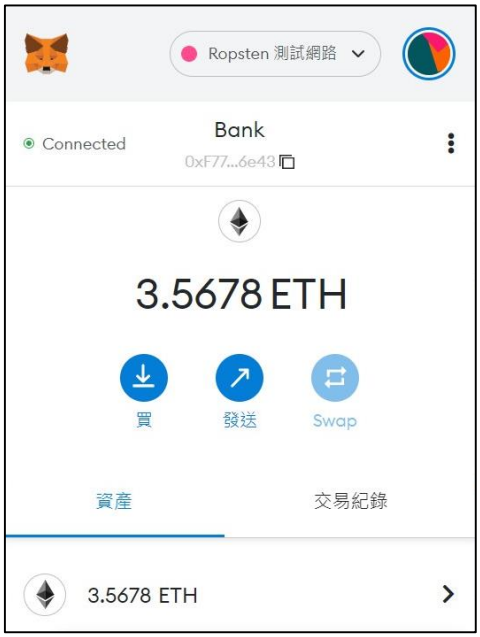




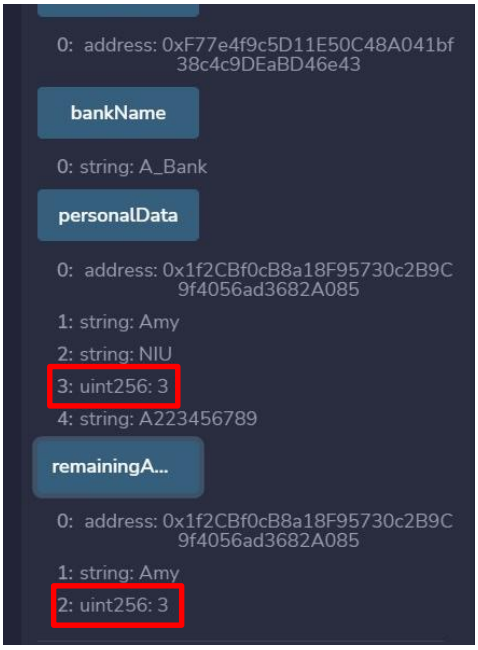
▲學生還款後，錢包金額之變化

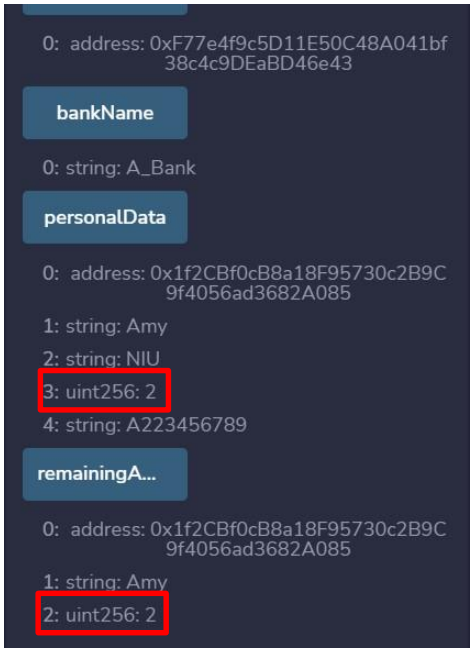
3.銀行收到學生償還的貸款





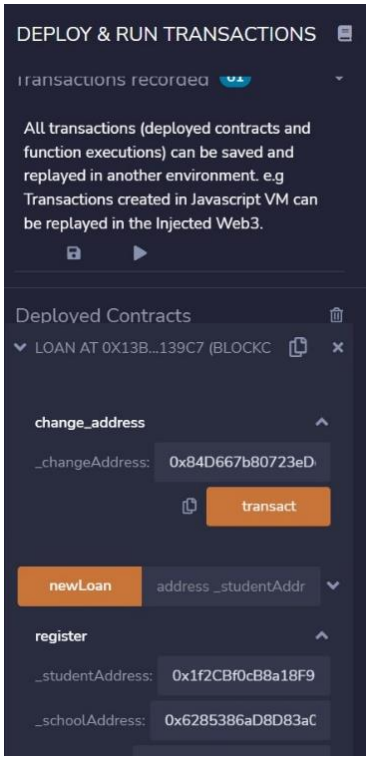
▲銀行收到學生還款後，錢包金額之變化

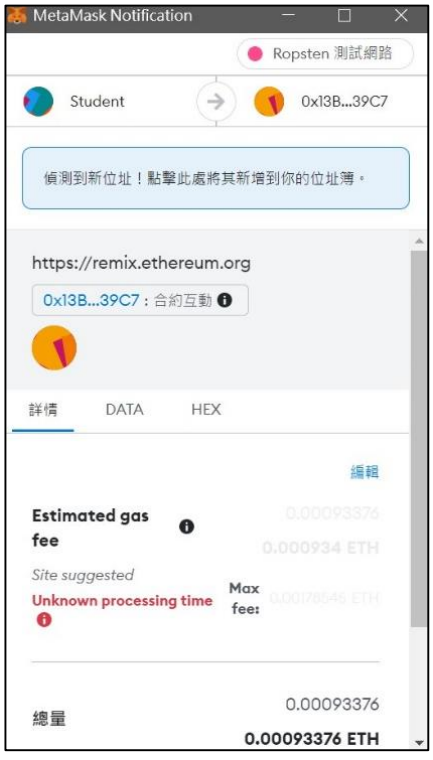




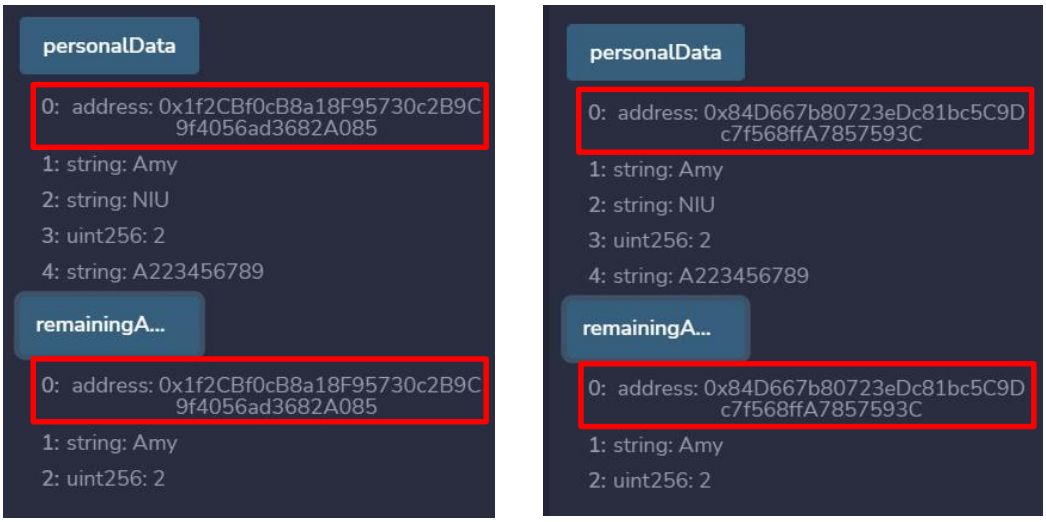
▲學生還款後，更改貸款的資料

4.更改錢包位址





▲學生更改錢包位址資料輸入，以及其手續費



▲學生更改錢包位置後，資訊也跟著改變