**Group**: Team 23
**Members**: Joshua Schmantowsky

# Application Requirements

## Functional Specification

The main purpose of the money manager app is to help people manage their money by keeping track of their daily transactions to make sure they are not overspending.

**Add Income and Savings**

1. The user opens the application.
2. The user selects the text box to the right of "Income:" and inputs their income
3. The user selects the text box to the right of "Savings:" and inputs their savings amount.
4. The system subtracts the savings amount from their income to determine their maximum budget.

**Changing Income or Savings**

1. The user wants to change their income or savings amount.
2. The user inputs new amounts for either the income, savings, or both.
3. The user hits the "Submit" button.
4. The application removes all previous transactions from the balance and clears the transaction table.

**Log a Transaction**

1. The user selects the text box to the right of "Transaction Name" and inputs the name of the transaction.
2. The user selects the text box to the right of "Transaction Amount" and inputs the total amount from the transaction.
3. The user selects one of the radio buttons to specify the type of transaction that has occurred.
4. The user selects the "Add" button to add the transaction.
5. The system displays the transaction on the home page and deducts the amount of the transaction from the balance.

**Remove a Transaction**

1. The user clicks on the transaction they would like to have deleted.
2. The user clicks the delete button to delete the transaction.
3. The system removes the transaction from the transaction total and adds the amount back to the balance.

**Determining Over Budget**

1. The user inputs their income and savings.
2. The user inputs all of their transactions.
3. As the user inputs their transactions, the balance will decrease.
4. If the system determines that the balance will go below zero, the system displays a notification to the user that they are over budget and must adjust.

**Viewing Graphical Amount**

1. The user opens the application, where a secondary window will pop up with a bar graph.
2. The user inputs their income and savings.
3. The user inputs their transactions.
4. As a transaction is saved or removed by a transaction type, the bar on the graph representing the transaction type adjusts accordingly.

| | |
|---|---|
| **Use Case:** | UC-01 |
| **Use Case Name:** | Money Manager/Expenses Tracker |
| **Created By:** | Joshua Schmantowsky |
| **Date Created:** | 2/13/2019 |
| **Preconditions:** | None |
| **Post-conditions:** | The user is able to track expenses and view summaries using the application. |

| **User Interaction** | **System Response** |
|---|---|
| Add Income | Add Income to the account |
| Add Savings | Subtracts savings from income for total balance |
| Add Transactions | Subtract leisure expenses from income |
| | Subtract bill expenses from income |
| | Subtract emergency expenses from income |
| | Subtract pet expenses from income |
| Add too many transactions or too high amount | Warn user they’re out of budget |
| | View graph of transactions |

|  |  |
|---|---|
| **Use Case:** | UC-02 |
| **Use Case Name:** | Add Income and Savings |

|  |  |
|---|---|
| **Description:** | The user is looking to add an income total to the application. |
| **Actors:** | User, Application |
| **Preconditions:** | 1. The user must have the application active. |
| **Post-Conditions:** | 1. The application now shows income, savings, and a balance for spending. |
| **Normal Flow:** | 1. The user selects the text box to the right of “Income” and inputs their income.<br>2. The user selects the text box to the right of “Savings” and inputs their savings amount.<br>3. The user selects the “Submit” button after inputting the savings and income.<br>4. The application determines the total balance available for use by subtracting the savings from the income.<br>5. The balance is displayed at the lower-right corner of the application. |
| **Alternate Flow:** | None |
| **Exceptions:** | 1. If the user inputs anything except for a number, the will be provided with an error stating an invalid number.<br>2. If the user inputs savings greater than the income, an error will display notifying the user that they cannot have a savings higher than the income and will not continue until this is changed.<br>3. If the user implements a number with more than two decimal points for both savings and income, the balance will round to the nearest hundredth decimal point. |

| | |
|---|---|
| **Use Case:** | UC-03 |
| **Use Case Name:** | Changing the Income or Savings |
| **Description:** | The user needs to change the savings or income amount. |
| **Actors:** | User |
| **Preconditions:** | 1. The application must be launched. |
| **Post-Conditions:** | 1. The user is able to add transactions in the application. |
| **Normal Flow:** | 1. The user inputs new amounts for either the income, savings, or both.<br>2. The user hits the “Submit” button.<br>3. The application removes all previous transactions from the balance and clears the transaction table. |
| **Alternate Flow:** | None |
| **Exceptions:** | 1. The user will need to redo all previously-submitted transactions. |

|  |  |
|---|---|
| **Use Case:** | UC-04 |
| **Use Case Name:** | Log a Transaction |

|  |  |
|---|---|
| **Description:** | The user needs add a transaction to the application. |
| **Actors:** | User |
| **Preconditions:** | 2. The application must be launched. |
| **Post-Conditions:** | 2. The user is now able to add transactions in the application. |
| **Normal Flow:** | 4. The user is shown their transactions on the main menu.<br>5. The user selects the text box next to “Transaction Name” and inputs the transaction name.<br>6. The user selects the text box next to “Transaction Amount” and inputs the amount.<br>7. The user selects one of the four radio buttons indicating the type of transaction it is.<br>8. The user selects the “Add” button.<br>9. The transaction is added to the table above.<br>10. The application adds the transaction to the specified transaction total.<br>11. The application subtracts the transaction from the total balance.<br>12. The application displays the new balance at the bottom-right of the main menu. |
| **Alternate Flow:** | None |
| **Exceptions:** | 2. If the user inputs anything but a number in the “Transaction Amount” textbox, they will receive an error indicating invalid characters. |

| | |
|---|---|
| **Use Case:** | UC-05 |
| **Use Case Name:** | Remove a Transaction |
| **Description:** | The user needs to remove a transaction from the transaction list. |
| **Actors:** | Users |
| **Preconditions:** | 1. The user is at the main menu. |
| **Post-Conditions:** | 1. The user now has additional money in their budget. |
| **Normal Flow:** | 1. The user selects the specified transaction from the transaction list.<br>2. The user selects the “Delete” button below the transaction list to delete the transaction.<br>3. The transaction is removed from the list.<br>4. The application subtracts the transaction from the specified transaction total.<br>5. The application adds the transaction to the total budget.<br>6. The new budget is displayed at the bottom-right side of the main screen. |
| **Alternate Flow:** | None |
| **Exceptions:** | 1. If there are no transactions in the table, the user will not be able to delete a transaction. |

|  |  |
|---|---|
| **Use Case:** | UC-06 |
| **Use Case Name:** | Determining Over Budget |

|  |  |
|---|---|
| **Description:** | The application will warn the user they are over their budget |
| **Actors:** | User, Application |
| **Preconditions:** | 1. The user is at the main menu<br>2. The user has already submitted their savings and income. |
| **Post-Conditions:** | 1. The user is notified if they are exceeding their set budget. |
| **Normal Flow:** | 1. The user adds transactions to the transactions table.<br>2. The system keeps log of the transactions that are being submitted and updates the balance accordingly.<br>3. If the balance gets lower than 0, a notification message will appear to notify the user they are over-budget, but will not stop them from submitting additional transactions. |
| **Alternate Flow:** | None |
| **Exceptions:** | None |

|  |  |
|---|---|
| **Use Case:** | UC-07 |
| **Use Case Name:** | Viewing Graphical Amount |
| **Description:** | The user would like to see a graphical representation of their transactions. |
| **Actors:** | User, Application |
| **Preconditions:** | None |
| **Post-Conditions:** | 1. The application shows a bar graph representing the transaction amounts for each type. |
| **Normal Flow:** | 1. The user opens the application, where a secondary window will pop up with a bar graph.<br>2. The user inputs their income and savings.<br>3. The user inputs their transactions.<br>4. As a transaction is saved or removed by a transaction type, the bar on the graph representing the type adjusts accordingly. |
| **Alternate Flow:** | None. |
| **Exceptions:** | None. |

# Project Design Specification

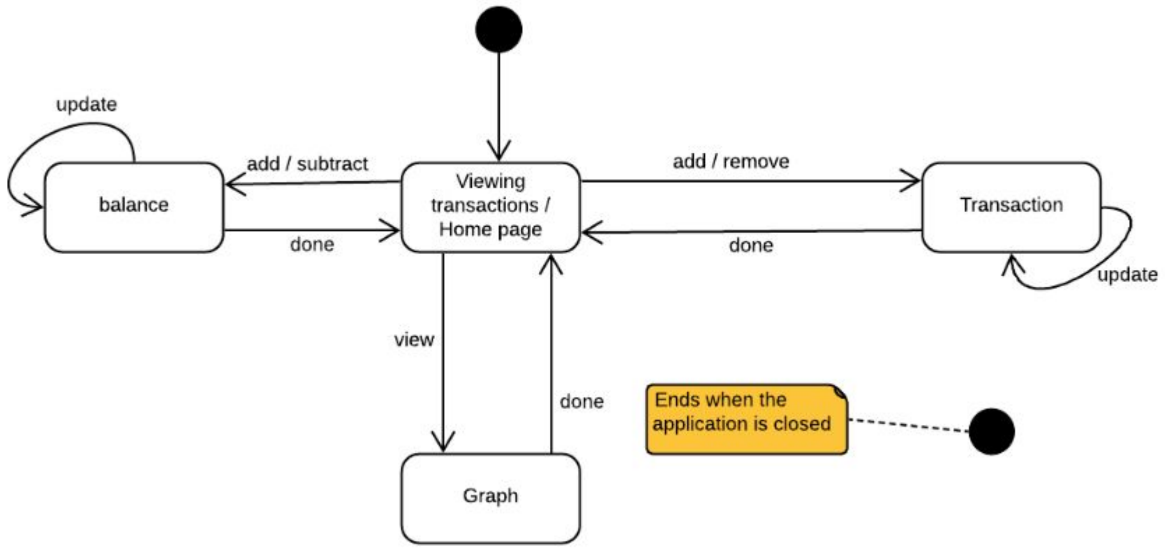
## CRC cards

| HomePageController | |
|---|---|
| Adds income | MoneyLog |
| Adds savings | |
| Displays transactions | |
| | |

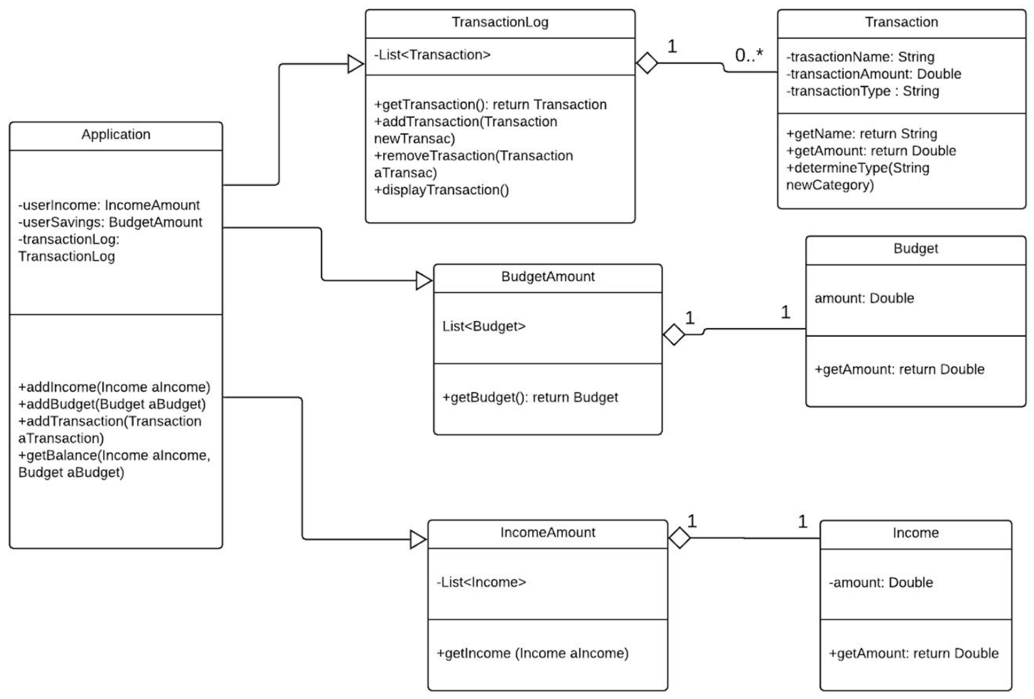
| TransactionLog | |
|---|---|
| Adds transactions | HomePageController |
| Remove transactions | TransactionGraph |
| Gets Balance | |
| | |

| TransactionGraph | |
|---|---|
| Show graph | MoneyLog |
| Update Graph | |
| | |
| | |

## State Diagram

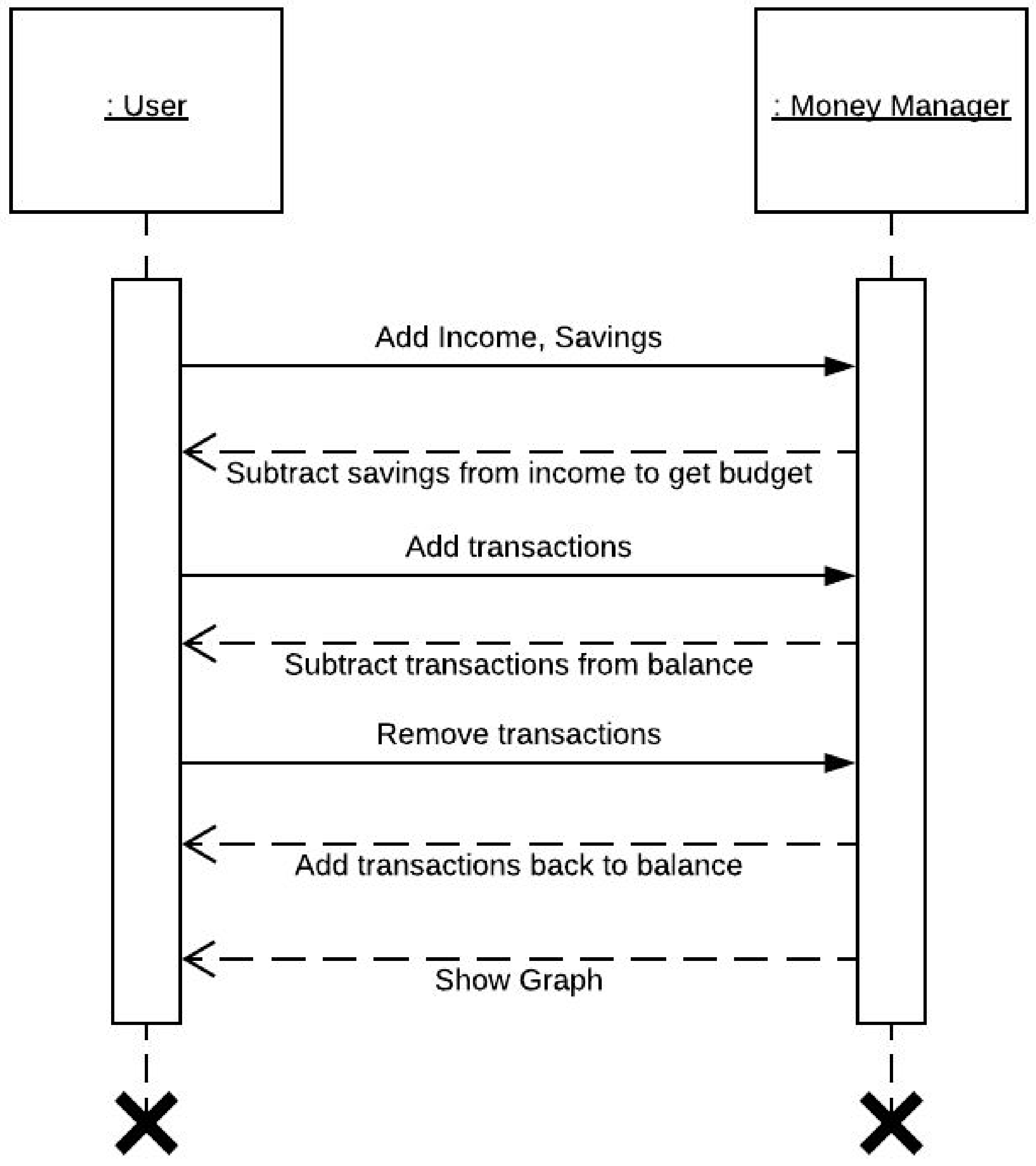


## Class Diagram

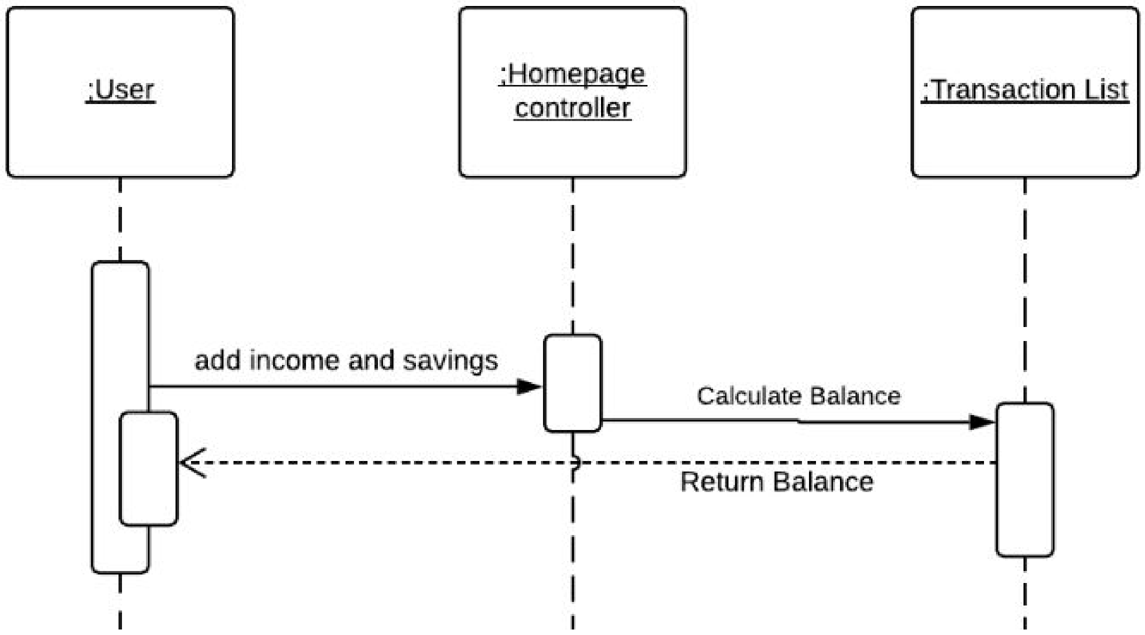

TransactionLog
-List<Transaction>
+getTransaction(): return Transaction
+addTransaction(Transaction newTransac)
+removeTrasaction(Transaction aTransac)
+displayTransaction()
1
0..*
Transaction
-trasactionName: String
-transactionAmount: Double
-transactionType : String
+getName: return String
+getAmount: return Double
+determineType(String newCategory)
Application
-userIncome: IncomeAmount
-userSavings: BudgetAmount
-transactionLog: TransactionLog
+addIncome(Income aIncome)
+addBudget(Budget aBudget)
+addTransaction(Transaction aTransaction)
+getBalance(Income aIncome, Budget aBudget)
BudgetAmount
List<Budget>
+getBudget(): return Budget
1
1
Budget
amount: Double
+getAmount: return Double
IncomeAmount
-List<Income>
+getIncome (Income aIncome)
1
1
Income
-amount: Double
+getAmount: return Double

## <u>Sequence Diagrams</u>

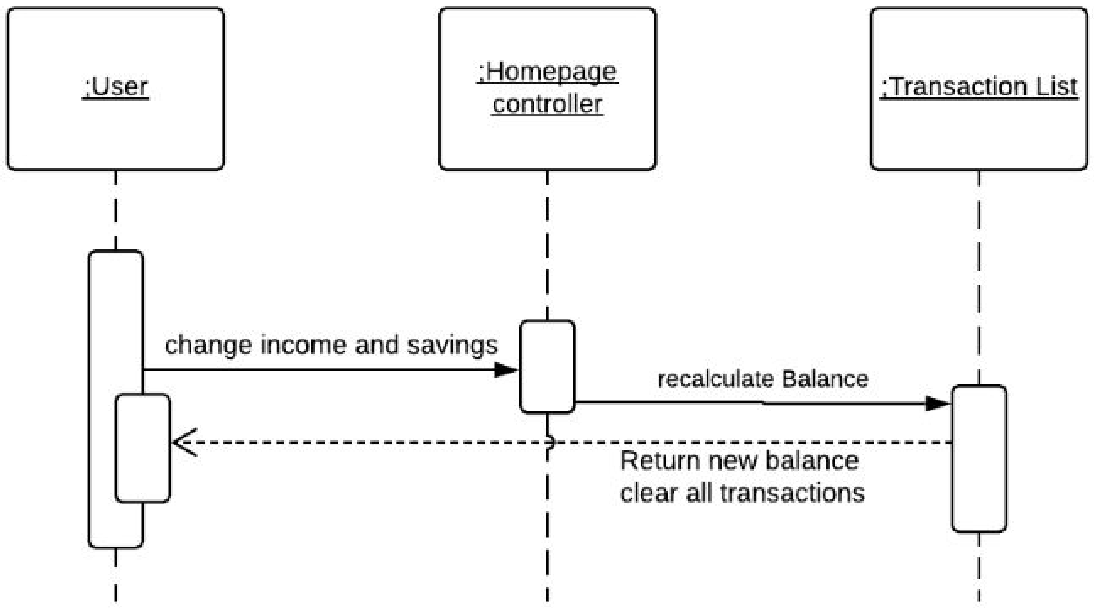
### Sequence 1 (overall)


: User
: Money Manager
Add Income, Savings
Subtract savings from income to get budget
Add transactions
Subtract transactions from balance
Remove transactions
Add transactions back to balance
Show Graph

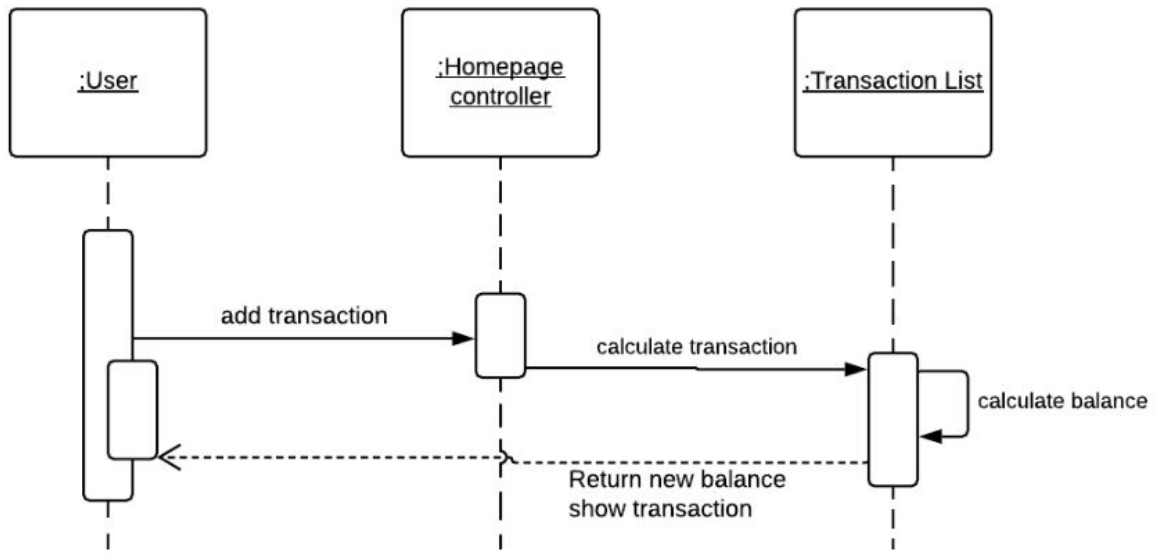
## Sequence 2 (add income and savings)



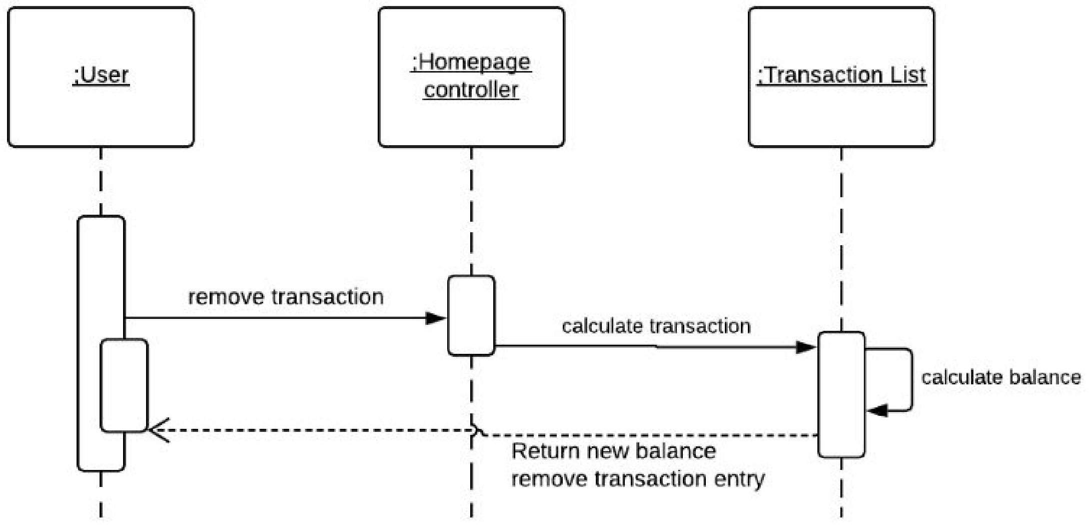
## Sequence 3 (change income or savings)



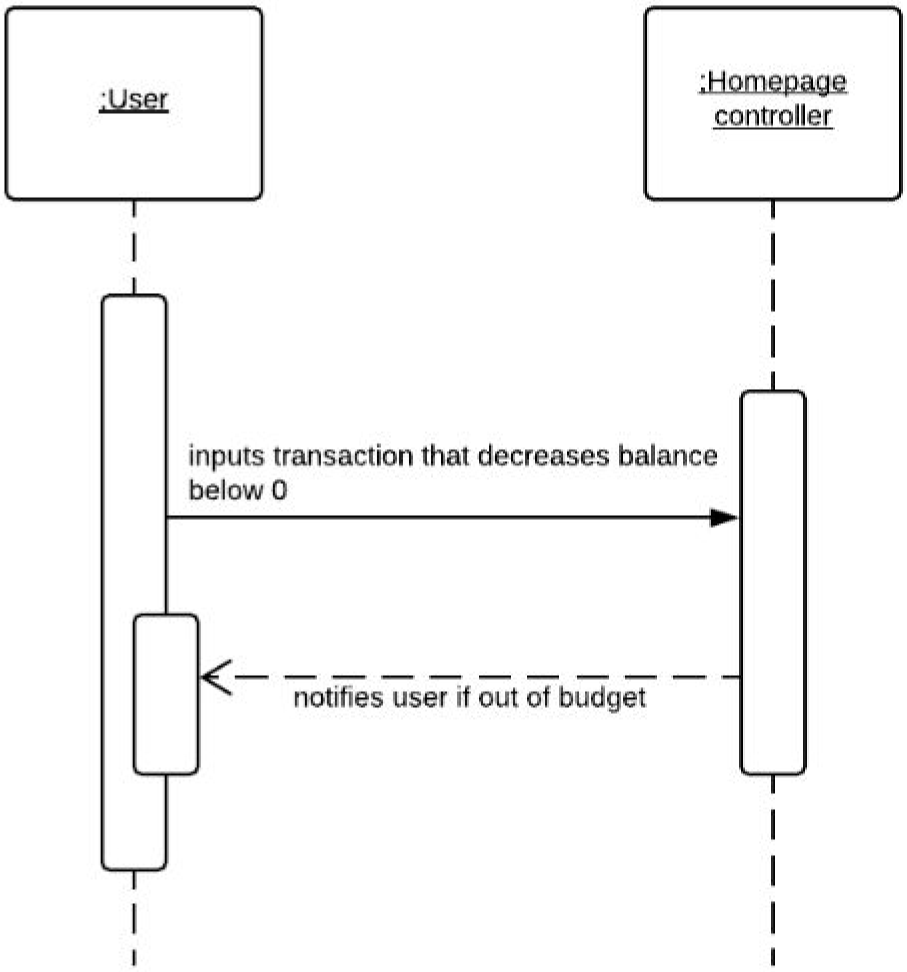
## Sequence 4 (log a transaction)



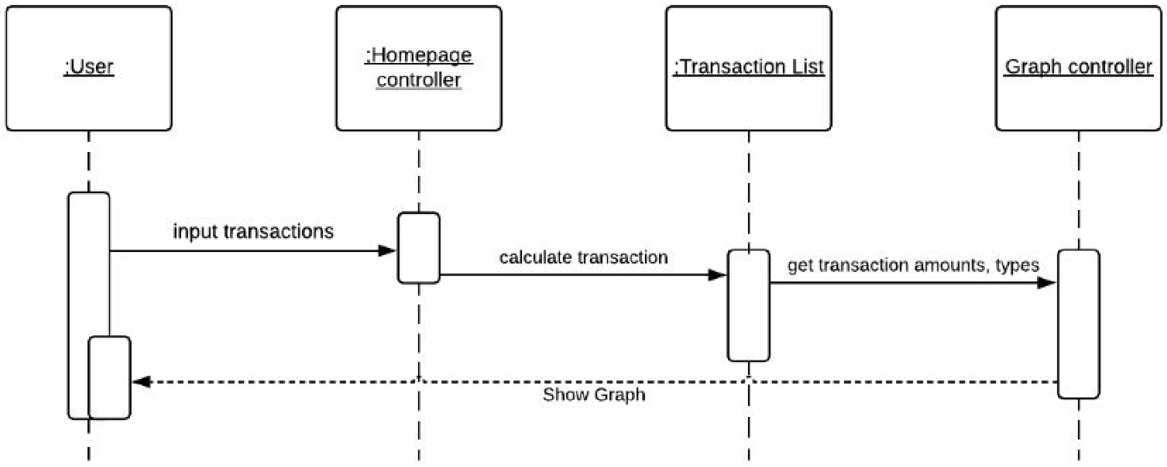
## Sequence 5 (remove a transaction)



## Sequence 6 (determine over budget)


:User
:Homepage controller
inputs transaction that decreases balance below 0
notifies user if out of budget

## Sequence 7 (show graph)



## Lessons Learned & Conclusion

This project was interesting for me to work on with the challenges that I faced throughout it. Initially I was set to work on it with a project partner, but due to a medical emergency I've had the opportunity to think about the project, reduce the scope, and still retain the goal: to create a money saving application that helps the user know when they're spending over-budget.

The biggest lesson I've learned from this project was understanding when the scope was too big. Initially, I thought I could implement more to the project to make it a more complete application, but I quickly realized that while it might have been feasible with multiple partners, I wouldn't have enough time to make the project that I envisioned. Another lesson that I learned was the necessity to understand the whole concept of the project before diving in. When my group partner, Freguens Mildort, was still with the group he introduced the idea for a money saving application. I didn't realize that my vision and his vision of what he was looking for in the project were two different things. As we started talking about the project, I was thinking of one way of implementing functionality while he was thinking differently. As we talked throughout the semester, we both started to understand what might work best and what might have problems in the future. Through the diagrams that were created, we were able to come up the way in which we could execute the creation of this application while still allowing us to implement our thoughts.

When I was informed that he had to step down from the class, I wanted to keep as much functionality that we previously planned, so I had to make some design changes to succeed. By utilizing the outline of the project that Freguens envisioned, I was able to create a version of that application that has most of the core functionality with my own interpretation.