**Simulation**

**Introduction**

The simulation feature in the website focuses on simulating for the member to be able to buy and sell an amount of a coin. Below, are the all the planning stages of the simulation part and detailed explanations of the thought process into this feature.

The section below is the first planning stage which consists of scenarios, user stories, use cases, use case diagrams and sequence diagrams. These tools aid when defining the requirements and the structure of the simulation feature.

Scenario 1: Members should be able to buy and sell an amount of coin

User Story 1:

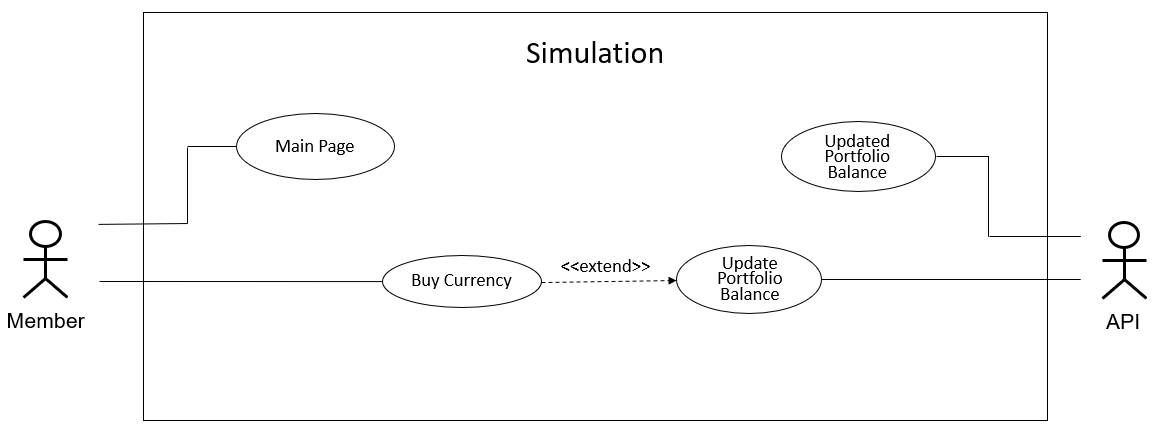
**Story:** As a member, I want to simulate buying an amount of coin so that I’m able to get a better understanding on buying any currency.

**Detail:** All member will have access to the main page, there will be a graph representing the price of bitcoin and options to buy/sell bitcoin with a text box to enter amount of bitcoin to be bought or sold. Before members start investing, they will first need to create a portfolio with a specified starting amount in the portfolio page. Once a portfolio is created, the members will then be able to analyse the graph representing the price of bitcoin and decide on an amount of bitcoin to buy. Once a member initiates the buy option, the account balance and the amount of bitcoin owned will be updated. Furthermore, there is no limit to how many coins members are able to buy as long as they have sufficient funds.

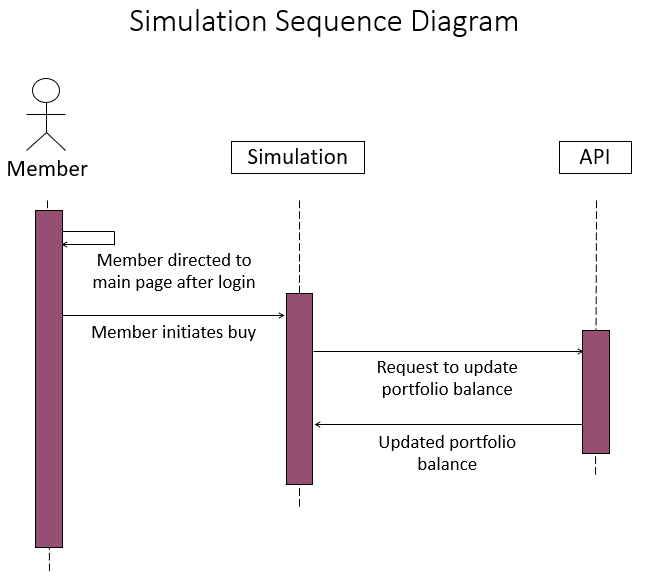
**Test:** When in the main page, check for sufficient funds, then in portfolio page check if there are portfolios created and check if a portfolio is selected. Check if when buy option is initiated, the portfolio balance and amount of bitcoin owned is updated.

UML Use Case Scenario:

|  |  |
| --- | --- |
| Use case name | Buy Currency |
| Participating actors | Member |
| Flow of events:  Normal flow | 1. Member is directed to main page after login 2. Member navigates to portfolio page to create a portfolio with a starting balance before buy 3. Member selects a portfolio 4. Member navigates back to main page 5. Member initiates buy 6. Portfolio balance and amount of currency owned is updated |
| Flow of events:  Alternative flow | Incorrect Data Format:   1. Member is directed to main page after login 2. Member navigates to portfolio page to create a portfolio with a starting balance before buy 3. Member selects a portfolio 4. Member navigates back to main page 5. Member initiates buy 6. Portfolio balance and amount of currency owned not updated due to incorrect data format entered   Insufficient Funds:   1. Member is directed to main page after login 2. Member navigates to portfolio page to create a portfolio with a starting balance before buy 3. Member selects a portfolio 4. Member navigates back to main page 5. Member initiates buy 6. Portfolio balance and amount of currency owned not updated due to insufficient funds |
| Pre-condition | Member hasn’t bought any currency |
| Post-condition | Member has bought currency  Member hasn’t bought any currency |

UML Use Case Diagram:

UML Sequence Diagram:



User Story 2:

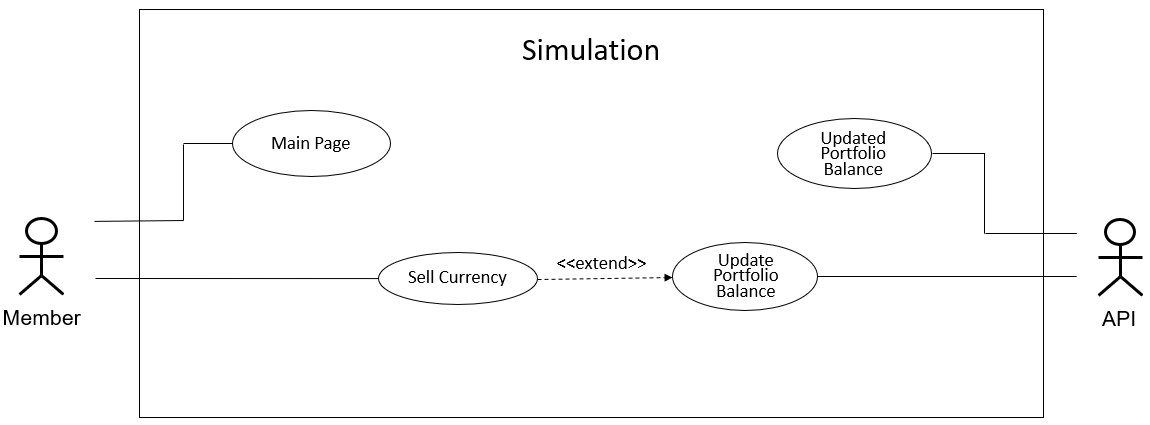
**Story:** As a member, I want to simulate selling an amount of coin so that I’m able to get a better understanding on selling any currency.

**Details:** All members will have access to the main page, there will be a graph representing the price of bitcoin and options to buy/sell bitcoin with a text box to enter amount of bitcoin to be bought or sold. Before members start investing, they will first need to create a portfolio with a specified starting amount in the portfolio page. Once a portfolio is created, the members will then be able to analyse the graph representing the price of bitcoin and decide on an amount of bitcoin to sell. However, members are only able to sell the bitcoins they own so if they down own any bitcoin, then the sell option will be hidden for them. Furthermore, members also cannot sell more than what they own. Once a member initiates the sell option, the account balance and the amount of bitcoin owned will be updated.

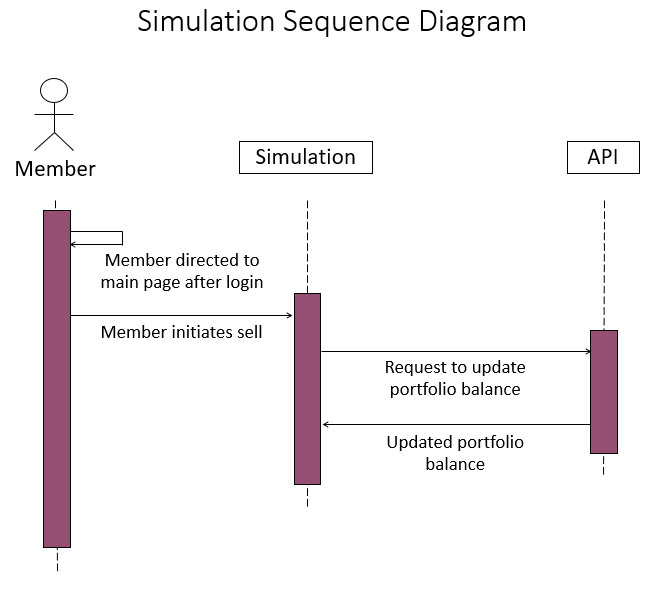
**Test:** When in the main page, check for sufficient funds, then in portfolio page check if there are portfolios created and check if a portfolio is selected. Check if when sell option is initiated, the portfolio balance and amount of bitcoin owned is updated.

UML Use Case Scenario:

|  |  |
| --- | --- |
| Use case name | Sell Currency |
| Participating actors | Member |
| Flow of events:  Normal flow | 1. Member is directed to main page after login 2. Member navigates to portfolio page to create a portfolio with a starting balance before sell 3. Member selects a portfolio 4. Member navigates back to main page 5. Member initiates sell 6. Portfolio balance and amount of currency owned is updated |
| Flow of events:  Alternative flow | Incorrect Data Format:   1. Member is directed to main page after login 2. Member navigates to portfolio page to create a portfolio with a starting balance before sell 3. Member selects a portfolio 4. Member navigates back to main page 5. Member initiates sell 6. Portfolio balance and amount of currency owned not updated due to incorrect data format entered   Insufficient Funds:   1. Member is directed to main page after login 2. Member navigates to portfolio page to create a portfolio with a starting balance before sell 3. Member selects a portfolio 4. Member navigates back to main page 5. Member initiates sell 6. Portfolio balance and amount of currency owned not updated due to insufficient funds |
| Pre-condition | Member hasn’t sold any currency |
| Post-condition | Member has sold currency  Member hasn’t sold any currency |

UML Use Case Diagram:

UML Sequence Diagram:



Scenario 2: Members should be able to view an updated transactions table with transactions associated to a selected portfolio

User Story 1:

**Story:** As a member, I want to be able to view my transaction’s history for a selected portfolio so I can track progress for that specific portfolio.

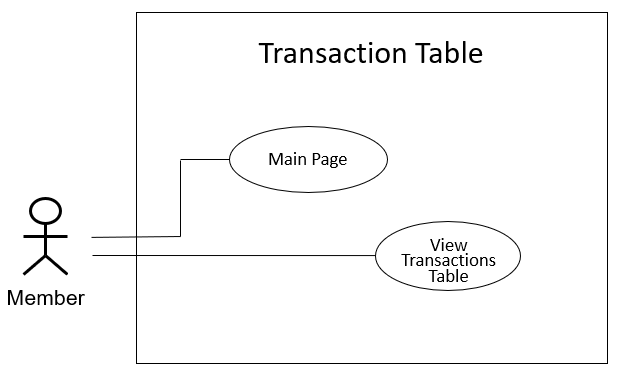
**Details:** All members will have access to the main page, there will be a graph representing the price of bitcoin and options to buy/sell bitcoin with a text box to enter amount of bitcoin to be bought or sold. There will also be a table displaying all transactions associated to a selected portfolio on the main page and every member will be able to view their own transactions every time one is made on the main page and keep updated.

**Test:** In the main page, check to see if there is a transactions table and that it shows the transactions made by member. Also check to see if data in table corresponds with the data entered.

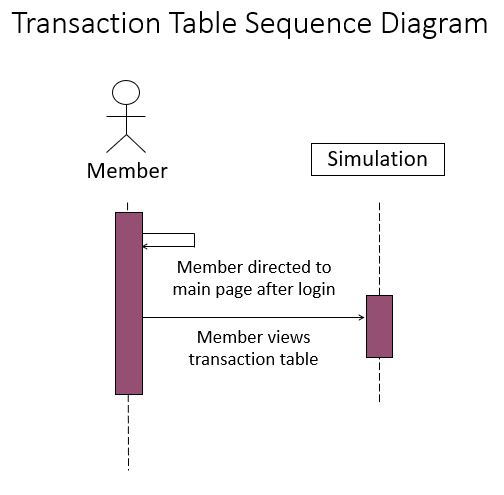
UML Use Case Scenario:

|  |  |
| --- | --- |
| Use case name | View Transaction Table |
| Participating actors | Member |
| Flow of events:  Normal flow | 1. Member is directed to main page after login 2. Member can view transactions table on main page |
| Flow of events:  Alternative flow | No Transactions:   1. Member is directed to main page after login 2. Member can not view transactions table as there are no transactions made |
| Pre-condition | Member hasn’t viewed transaction table |
| Post-condition | Member has viewed transaction table  Member hasn’t viewed transaction table |

UML Use Case Diagram:



UML Sequence Diagram:



User Story 2:

**Story:** As a member, I want to be able to view an updated table of my transaction’s history for a selected portfolio so I can track progress for that specific portfolio.

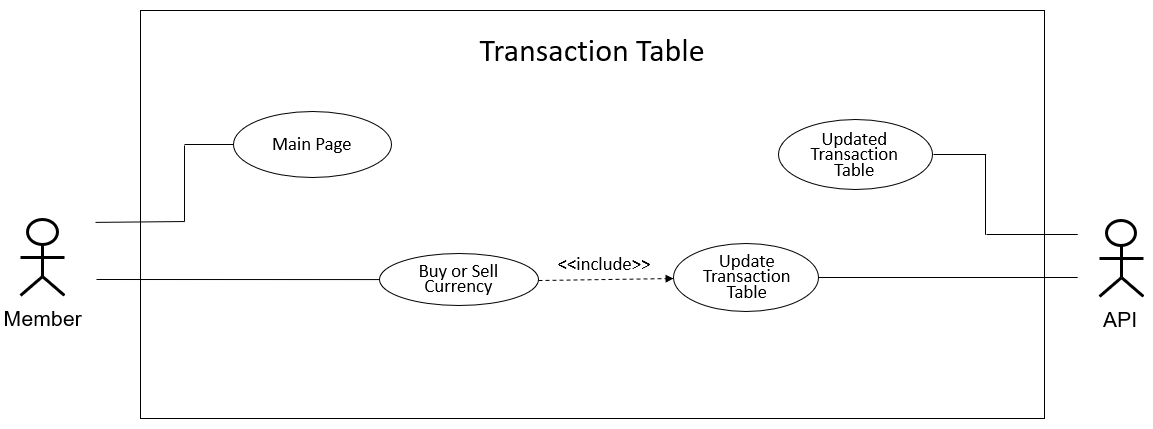
**Details:** All member will have access to the main page, there will be a graph representing the price of bitcoin and options to buy/sell bitcoin with a text box to enter amount of bitcoin to be bought or sold. There will also be a table displaying all transactions associated to a selected portfolio on the main page. Once a coin is bought or an amount of a coin is sold, then the transactions table will be updated with the transaction type (Bought/Sold), amount bought or sold and the time the member bought or sold the coin. Every member will be able to view their own transactions associated to a selected portfolio on the main page and keep updated.

**Test:** In the main page, check to see if there is a transactions table and that it is updated every time a transaction is made. Also check to see if updated data in the table corresponds with the data entered.

UML Use Case Scenario:

|  |  |
| --- | --- |
| Use case name | Update Transaction Table |
| Participating actors | API |
| Flow of events:  Normal flow | 1. Member is directed to main page after login 2. Member navigates to portfolio page to create a portfolio with a starting balance before buy or sell 3. Member selects a portfolio 4. Member navigates back to main page 5. Member initiates buy or sell 6. Transaction table is updated with the currency bought/sold, the amount bought/sold, the transaction type (if currency is bought or sold) and the time the currency is bought at |
| Flow of events:  Alternative flow | Incorrect Data Format:   1. Member is directed to main page after login 2. Member navigates to portfolio page to create a portfolio with a starting balance before buy or sell 3. Member selects a portfolio 4. Member navigates back to main page 5. Member initiates buy or sell 6. Transaction table is not updated with the currency bought/sold, the amount bought/sold, the transaction type (if currency is bought or sold) and the time the currency is bought at due to incorrect data format entered   Insufficient Funds:   1. Member is directed to main page after login 2. Member navigates to portfolio page to create a portfolio with a starting balance before buy or sell 3. Member selects a portfolio 4. Member navigates back to main page 5. Member initiates buy or sell 6. Transaction table is not updated with the currency bought/sold, the amount bought/sold, the transaction type (if currency is bought or sold) and the time the currency is bought at due to insufficient funds |
| Pre-condition | Transaction table not updated |
| Post-condition | Transaction table updated  Transaction table not updated |

UML Use Case Diagram:



UML Sequence Diagram:

