**Simulation Research**

**Introduction**

The simulation feature allows members to simulate buy/sell options. The sole purpose of this research is to get a better understanding on the steps that need to be taken to implement this feature.

Firstly, I started mapping out roughly the steps I would need to take during the implementation stage, shown below:

* Azure Table Storage to store currency data
* API (POST end-point) - add endpoint to send data to table storage
* API (GET end-point) - add endpoint to get data from table storage
* UI option to buy or sell a currency
* UI option to enter amount of currency to be bought or sold
* UI table to display member transactions
* Updating UI transactions table and portfolio data when buy or sell option is initiated

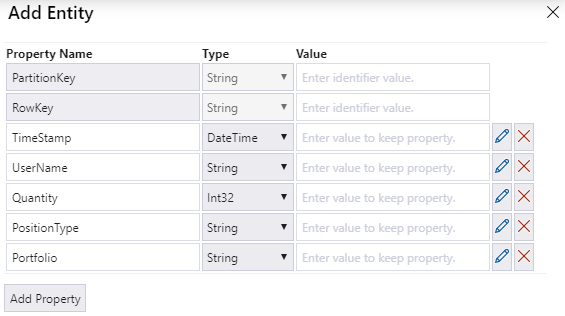
I began by researching into azure as I needed to create a storage table to store currency data. The link below gives an overview of how to create storage tables in azure also, providing background information about what azure storage table storage is and what it does.

Overview of Azure Table Storage:

<https://www.youtube.com/watch?v=HSL1poL1VR0>

After creating the storage table, I had to think about what entities I want to storage about each currency. I came to the conclusion of storing the username which would represent who is making the transaction, a timestamp to represent when the transaction is made, the quantity of currency to represent amount of currency bought or sold, the positions type to represent if currency is bought or sold and portfolio to represent which portfolio the transaction is made in. Figure 1 is a screenshot of me adding the entities to the storage table.

Figure 1:



The next stage in implementation was to create an API and add end points the send data to and get data from the storage table I created. In this stage I got help from my group member Dexter with sending data to the storage table and also followed the links below to retrieve data from the storage table.

Retrieving Data:

<https://www.youtube.com/watch?v=z96fIv3RQBo&t=191s>

<https://docs.microsoft.com/en-us/rest/api/storageservices/writing-linq-queries-against-the-table-service>

Once endpoints are created, it was time for testing. As a group, we used two applications for testing which consist of Fiddler and Postman. I decided to use Fiddler to test my endpoints. My group member Dexter provided guidance once more with understand how to use Fiddler and how to test endpoints locally.

The next stage was to hook up endpoint to the front end and as we were using the Angular framework, I took some time understanding how Angular works. The links below helped understand the fundamentals when designing the UI of a website through Angular.

Angular Tutorial:

<https://www.youtube.com/watch?v=k5E2AVpwsko>

Angular Guide:

<https://angular.io/guide/architecture>

Displaying Data in Angular:

<https://angular.io/guide/displaying-data>

During the implementation of the front end. It was a struggle to call the API endpoint and get data on to the front end. The help of the links below guided me with this section.

Angular API Calls:

<https://www.youtube.com/watch?v=_05v0mrNLh0>

<https://www.youtube.com/watch?v=rdLJNGZvlAA&t=525s>

For the implementation of the transactions table in the website. I research about the scrollbar feature as I thought it would be good to implement so transactions don’t overflow in the page. The link below shows guidance to implement a scrollbar feature.

Scrollbar in Angular:

<https://material.angular.io/cdk/scrolling/overview>