NextGen / NextGen Stock Market Game

**1. The Team**

|  |  |
| --- | --- |
| Student Number | Name |
| 16211140 | Salmaan I M M |
| 16211146 | Samaranayaka W T M |
| 16211258 | Botejue J W I A |
| 16211257 | Wickramathilaka K B K |
| 16211226 | Sandaruwan M K D A |
| 16211262 | Makawita M A C D |

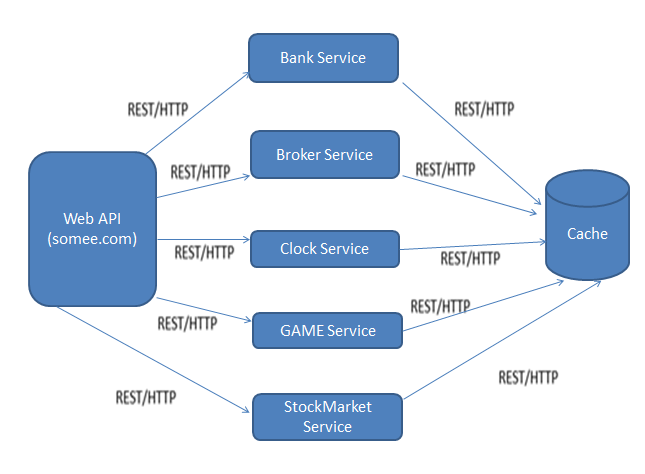
|  |  |
| --- | --- |
| *Repository Type* | GitHub |
| *Repository URL* | https://github.com/salmaanzats/NextGenStockMarket |
| *Kanban Board URL* | https://github.com/salmaanzats/NextGenStockMarket/projects/1 |
| *API Host URL* | http://nextgenstocksimulationapi.somee.com/ |

|  |  |
| --- | --- |
| **TASK** | **ASSIGNED TO** |
| Show the information required by player through GUI |  |
| Create the gaming strategy |  |
| Implementing the simulator Architecture |  |
| Create an bank account |  |
| Implementing the simulator frontend Architecture |  |
| Implementing the Broker service - create account |  |
| Implementing the Broker service - get companies |  |
| Create a method to start a new game |  |
| Create a method to get winner |  |
| Create a GUI for player registration |  |
| Create a GUI for sign in |  |
| Add multiple players for the game(Create a method to connect players) |  |
| Create a method to get the status of the game |  |
| Create architecture of the game (Change initial web page like architecture to a game) |  |
| Create player profile interface |  |
| Implementing the Broker service - sell |  |
| Implementing the Broker service -get sectors |  |
| Implementing the Broker service - buy |  |
| Create side bar access in GUI |  |
| Make the online compatible game |  |
| Prepare documentation |  |

**2. Technology Choices**

|  |  |
| --- | --- |
| Technology | Motivation for Use |
| Caching | Increase the performance. Instead of repeatingly calculating an operation using a database, we were able to cache the result/data in a lesser time. |
| REST | All the team members are familiar with REST. |
| Angular | Easy to build applications with the web |
| Asynchronous Programming | Allow us to apply Parallel programming. |
| .NET Framework | Provides a platform for Caching, Asynchronous Programming, and MultiThreading. |

**3. System Architecture**



*Mainly there’re 7 components in our game. (Web API, Bank Service, Broker Service, Clock Service, Game Service, StockMarket Service & Cache)*

*While Web API is hosted on somee.com our project includes 5 main services such as*

* + *Bank Service (Manage users bank accounts, & all the financial activities related to purchase /sell),*
  + *Broker Service (Gives information/Provide help to users to buy/sell stocks),*
  + *Clock Service (A shared component which provides the current turn number),*
  + *Game Service (This includes the game structure with 4 sectors & 3 stocks),*
  + *StockMarket Service (This includes the Stock Value Changing Algorithm which calculates game score, assigning probabilities to events such as BOOM,BUST,PROFIT\_WARNING,TAKE\_OVER & SCANDAL )*

*& instead of a database we used Caching concept to handle data according to users.*

**4. Reflections**

* *Were the technologies appropriate?*

*Yes*

*Using angular it was easy to build the frontend of our game.*

*.net framework provides a great platform for features like Caching,*Asynchronous Programming, and MultiThreading which was appropriate for the tasks like, easy to divide the tasks along members, improve the performance of our game, easy to handle changeable data, etc.

*REST support the communication between users & the web service.*

* *What were the most difficult parts of the project?*

*Developing the Stock value change algorithm was quite a challenge for us. Mostly we search about setting priority for events based on probability through Google. We had to clear so much errors when we were implementing the algorithm. Actually most errors were came from the algorithm. So it was quite a hard task but mostly it was a great experience. And for some of our team members, starting with Angular was a challenge since they hadn’t got the experience on Angular at the start of our project. So they had to go through a lot of tutorials on Angular.*

* *What are the limitations of the technology you used?*

*Gaming platform in Angular is not totally stable*

*Game is going for 12 turns for 4 players.*

* *What are the benefits of the technology you used?*

*With the use of* Asynchronous Programmingwe were able to get the advantages of Parallel Programming.

*And when it comes to REST APIs they are stateless, meaning that calls can be made independently of one another. While stateless APIs handles large amount of incoming & outbound calls, we used caching to design the API to use the cacheable data (instead of a database) which increase the performance of our game.*

*While Angular is built on component based architecture, therefore it was easy to assigned the tasks to members & easy to separate the features.*

* *What did you learn?*

*Especially we learnt about working as a team through GitHub. (How to commit, pull a request, merge pull requests, etc). Other thing is Angular, Some members of our team didn’t had the knowledge on Angular at the beginning of our project. So they had to self-study about it from Google. And the other one is Caching. In our previous projects we always used Databases to handle data, but here instead of databases we all learnt about the Caching feature(how to derive data from cache, how caching works with Asynchronous programming cocepts,multi-threading,etc.)*

*Overall we all learnt about going through a big project with a big scope as a team.*

**References**

[1 Components of a stock market, <https://en.wikipedia.org/wiki/Stock_market>

[2]Angular tutorial, <https://angular.io/tutorial>

[3] Article about Asynchronous Programming in visualstudiomagazine.com, <https://visualstudiomagazine.com/articles/2011/03/24/wccsp_asynchronous-programming.aspx?m=1>