BLACKLAMPS

**1. The Team**

*Give details of your team members and their assigned tasks below.*

|  |  |
| --- | --- |
| Student Number | Name |
| 16211252 | R.M.B. Senanayake |
| 15209358 | R.M.A.Pawan.Sandaru |
| 16211291 | E.M.M. Ekanayake |
| 16211274 | D.U.C. Ranathunga |
| 16211115 | A.P.P. Perera |
| 16211409 | R.P.V.L. Madusanka |

*Provide the link to your repository. I must be given access to the code, the issues & the Kanban board that you used for the project. For GitHub, my username is “remcollier” while for GitLab, my username is “rem.collier”*

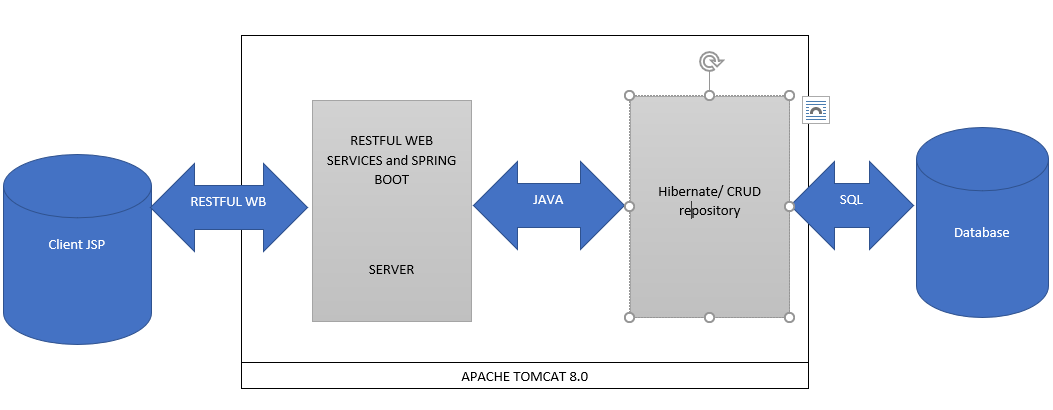
|  |  |
| --- | --- |
| *Repository Type* | GitHub |
| *Repository URL* | https://github.com/BlacklampsSE2/Game-Simulator |

**2. Technology Choices**

|  |  |
| --- | --- |
| Technology | Motivation for Use |
| Restful | Restful Web Services plays a huge role in the Game. All the communications between server and client are developed using restful web services. |
| Hibernate | We have used Hibernate framework for mapping an [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) domain model to a [relational database](https://en.wikipedia.org/wiki/Relational_database). Hibernate handles [object-relational impedance mismatch](https://en.wikipedia.org/wiki/Object-relational_impedance_mismatch) problems by replacing direct, [persistent](https://en.wikipedia.org/wiki/Persistence_(computer_science)) database accesses with high-level object handling functions. |
| CRUD Repository | Create, read, update, and delete are the four basic functions of [persistent storage](https://en.wikipedia.org/wiki/Persistent_storage). Alternate words are sometimes used when defining the four basic functions of *CRUD*, such as *retrieve* instead of *read*, *modify* instead of *update*, or *destroy* instead of *delete*. |
| Spring Boot | We have used Spring Boot reference implementation (https://spring.io/) to simplify the development of the web services. In order to develop events that needs to be broadcasted are developed using Spring Boot’s Server Sent Event which allows clients to subscribe to the events that originated on the server and clients to read SSE events asynchronously. (https://spring.io/documentation/latest/sse.html) |
| Spring Session | Can achieve session handling easily and can identify individual users uniquely. Easy to use. |

**3. System Architecture (max. 1 page)**

*Draw a system architecture diagram here that identifies all the key components of your system and how they interact.*



*Explain how your system works (in a broad sense) based on the architecture diagram you have given.*

*NOTE: On windows, I typically use PowerPoint to create diagrams and snipping tool to cut and paste the diagram into the document (see crappy example above).*

*The client loads the login page by using the URL. And If the client has been registered before the client can enter the credentials or can be registered as a new client. Upon both scenarios the credentials are passed to the server and the server authenticates it against the Database by querying it or it checks for the existing credentials and if it is not available it registers the user. All these communications are done using RESTFUL web services and the data is sent in POST method. Upon successful logon the server returns the index page (Main Menu) to the client.*

*Then upon clients wish the client can navigate to the GAME section and click on the PLAY NOW button. Upon this event the client sends and request to the server. The server checks the no. of players who are willing to play and if play count is three it returns the GAMEBOARD page to all the clients, otherwise it automatically assigns AI players and return the GAMEBOARD to the human client. To monitor all clients, we have used spring sessions which stores a cookie in the client side with the Session ID of the client which allows us to identify the client uniquely and display relevant details (Portfolio, Game History, etc). During the game the ANALYST sends the news feed and the rumours using a technology known as SERVER SENT EVENTS, it doesn’t need a client request to send the data to the client. Each turn is composed of 20 seconds the server sends data every 20 seconds to the client.*

*At the end of the game the winner is displayed and then the server returns all the clients to the index page (Main Menu).*

**4. Reflections (1 page)**

*Discuss (as a group) how well your project worked:*

* *Were the technologies appropriate?*
* *What were the most difficult parts of the project?*
* *What are the limitations of the technology you used?*
* *What are the benefits of the technology you used?*
* *What did you learn?*

*NOTE: This is a very important section, and you need good insights to achieve the A grade for the report quality category.*

* *For the communication between Client and Server we have used RESTFUL WEB SERVICES. And we have used CRUD REPOSIRTORY to make Database transactions more easy and reliable. SERVER SENT EVENTS have been use to send all data to client without any request from client, this made us reduce the effort on the server since client is not sending requests on each turn the data is automatically sent using SSE. We used spring boot to develop our application and MAVEN since it makes the development of the application so simple and easy to use and it can be integrated easily with other technologies. Spring sessions were used to identify clients and send the relevant data to them.*
* *The most difficult part was to load the data to the gameboard table and refresh the data every 20 seconds/turn to all the clients at the same time so that it is fair. And we faced a difficulty in Server sent events, when to send the data it shouldn’t be continuously, we tackled this issue by using threading. To program the AI player to make him random decisions according to the situations we achieved this after a struggle. Creating 10 company objects and making their stock price unique to each other. And handling the player buy/sell option in the gameboard company details table.*
* *When we used Server sent events we faced an issue when we integrated it with thread the response was timed out since it the response was delayed for 20 seconds/turn. We could use SQL queries to check whether the credentials user entered are correct so we had to use the CRUD repository we wrote a function do achieve this scenario.*
* *By using SSE, the load which comes to the server was reduced, client does not need to send requests each and every turn to get the data. By using CRUD repository to manage Database transactions it made it easy since we had to use the inbuilt CRUD functions to INSERT UPDATE so this made a real benefit for us. By using spring boot maven project, it made us easier to integrate with other technologies and spring made our activities much easier since it has embedded tomcat servers, automatically configure Spring and 3rd party libraries whenever possible, Absolutely no code generation and no requirement for XML configuration etc.*
* *This project was really valuable for us since we have never used these technologies before. We learnt about Spring boot and Session handling, SSE and CRUD repository. How to manage the transactions between the Client server and the database. We have never used RESTFUL web services before so we had to learn from the scratch. It was a real advantage in learning and then implementing these technologies for our project.*

# References

daneden. (n.d.). *A cross-browser library of CSS animations. As easy to use as an easy thing.* Retrieved from Github: https://github.com/daneden/animate.css

Edwards, T. (n.d.). *Sweet alerts*. Retrieved from sweetalert.js: https://sweetalert.js.org/guides/#installation

hatake, k. (n.d.). *Spring boot jpa crudrepository nullpointerexception to save*. Retrieved from stackoverflow.com: https://stackoverflow.com/questions/32571925/spring-boot-jpa-crudrepository-nullpointerexception-to-save

Paraschiv, E. (2018, May 6). *Configure a Spring Boot Web Application*. Retrieved from baeldung.com: http://www.baeldung.com/spring-boot-application-configuration

Phillip Webb, D. S. (n.d.). *Spring Boot Reference Guide*. Retrieved from spring.io: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

*Spring MVC - HTTP Streaming With Server-Sent Events*. (2017, Jan 4). Retrieved from logicbig: https://www.logicbig.com/tutorials/spring-framework/spring-web-mvc/sse-emitter.html

zaneacademy. (2014, Jan 21). Retrieved from Youtube: https://youtu.be/rhCKCKD0KnA?list=PLSM8fkP9ppPo4aE3Far8c8FC4fkxWHF8w

*You can reference any resources that you used, this can include web pages for technologies, user guides, …*

**Spring Boot**