Project 6

Group member names

Gregorio Figueroa
Darrien Lee
Jackson Meyer

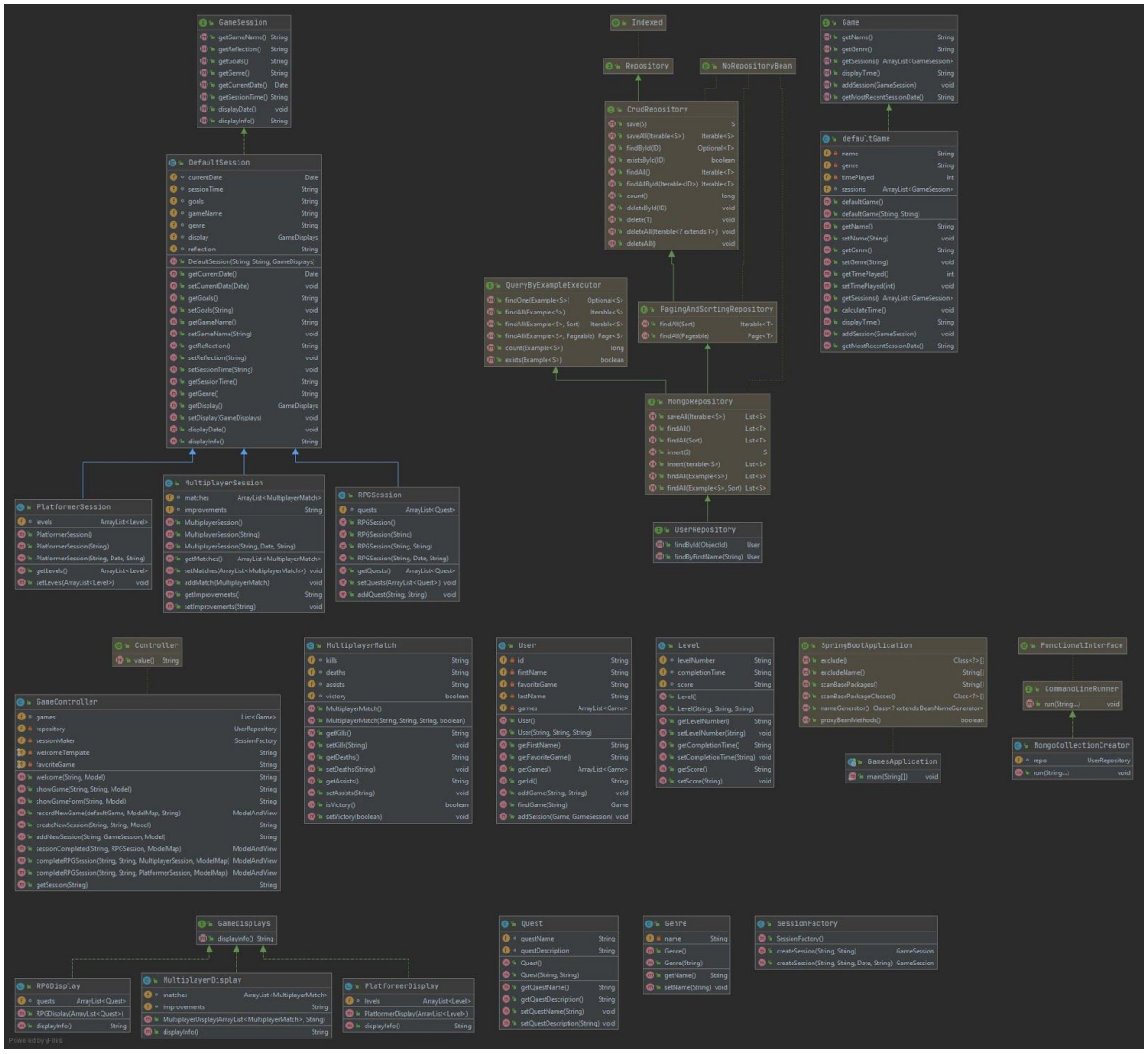
Final state of System

We would describe the final state of the system as a success. The system's use cases have been realized and everything has been expanded upon since the project 5 state. A user is able to create game sessions for any type of game and record unique information for a session, such as the amount of time or goals from the session. They are also able to see these sessions and create new ones for a specific home page for each game. We also added css and js elements to the project to expand on our webpage visuals and make it presentable. Some stretch features such as cloud based storage and user log in were not realized due to time. As these were not the main focus of the original project scope, we didn't worry about getting these stretch goals completed. Another feature that was a stretch goal that we didn't get to was updating and editing session information. Also, we wanted to implement the ability to give each session a time limit, but we ran out of time. Every other feature such as local storage, creating sessions and viewing game homepages was implemented and functions well.

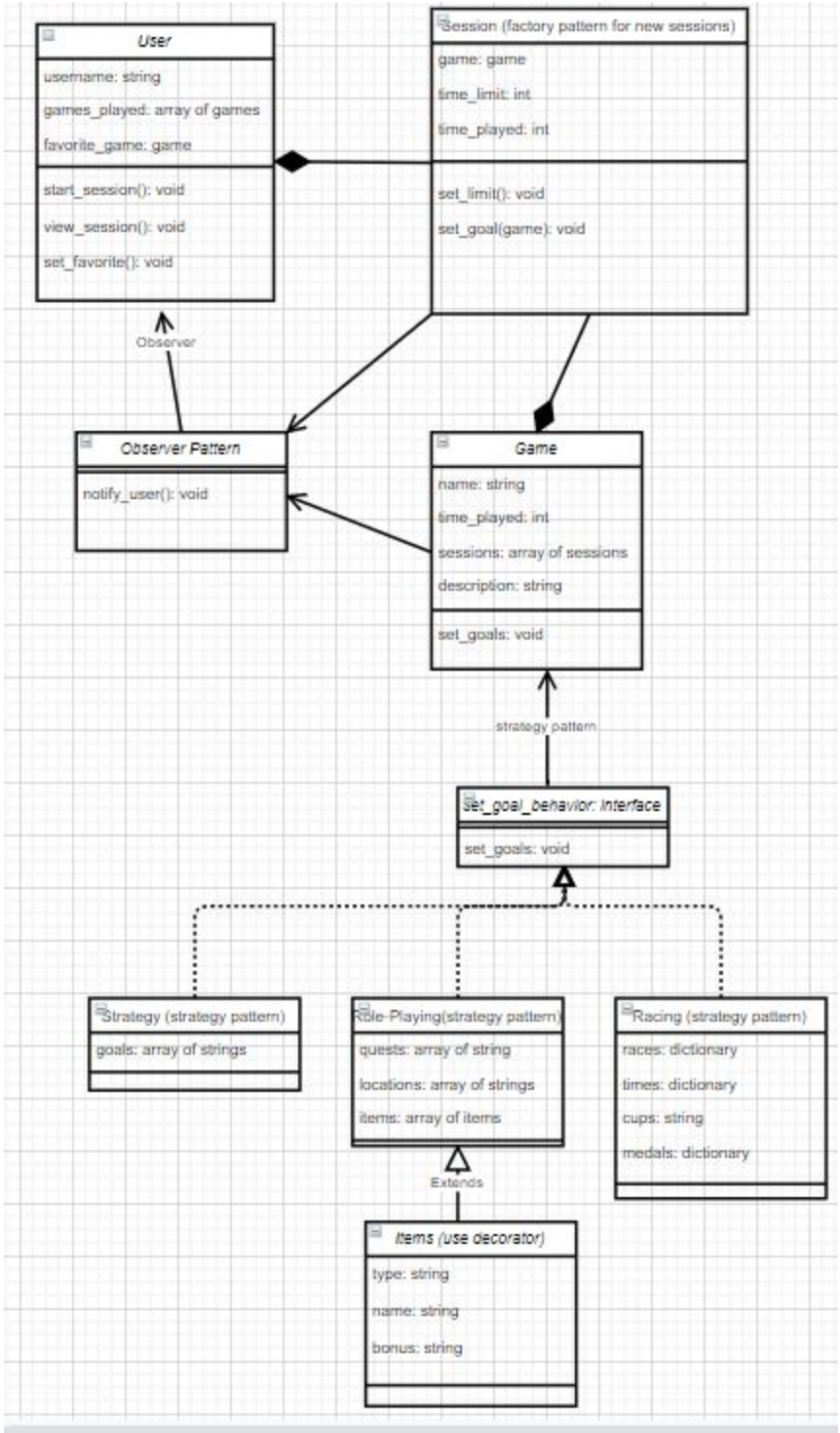
Class diagram

The class diagram from project 5 varied a lot from the project 4 class diagram. This was because early on we pivoted from using Node Js as our web framework to using Spring Boot. This caused the class diagram to be much different. The diagram has just been expanded upon to have session information and related classes like quests and levels. We also added the display classes to implement a simple strategy pattern on the display behavior for the classes. Overall the final class diagram is just an expanded version of the class diagram from project 5, and both are very different from the project 4 class diagram. We implemented the MVC pattern as the overarching pattern in our program, and implemented strategy and factory on top to make it easier to create new sessions (factory) and control display behavior (strategy pattern).

Project 6 UML Diagram



Project 4 UML Diagram



Our class diagram got more complicated as the project went along. These two UML class diagrams reflect how we changed our framework from NodeJS to Spring framework, and the differences that come along with it. Our new project 6 UML class diagram includes some helper classes that we weren't anticipating needing and shows that our program is now heavily based on the MVC pattern whereas the old project 4 UML class diagram did not implement MVC pattern.

Third Party Code Vs Original Code

All of the written code is located in the src/main folder. All the rest of code is generated by the spring initializer with specific dependencies. As far as the code for the controller and templates goes, most of it is original. We only used tutorials to know how to use spring and how to write the controller class, as well as with the timer. Most of the controller and thymeleaf code was based on the tutorials found at **https://spring.io/** . Other Resources are as follows:

Lists in Thymeleaf - https://www.baeldung.com/thymeleaf-list
Thymeleaf Introduction - https://www.thymeleaf.org/doc/tutorials/2.1/usingthymeleaf.html#fixed-value-boolean-attributes
MongoCrud help - https://bezkoder.com/spring-boot-mongodb-crud/
CSS documentation - https://www.w3schools.com/css/default.asp
Timer Example - https://stackoverflow.com/questions/5517597/plain-count-up-timer-in-javascript

OOAD Process

Wow this has been such a great experience learning more about OO analysis and design. Our group as a whole learned a lot this whole semester. There is a lot of good I can say about this whole process and OOAD has allowed us to work efficiently and with a clear purpose.

A key element to the success of this project was designing use cases and really thinking about the architecture before writing code. By writing the use cases and thinking about architecture, we were able to quickly begin on writing a base working version of the application and expand from there. Not once did we not know what to do next to proceed with development. We knew what we wanted from the finished project and it was just a question of how to do it. We referred back to use cases and initial sketches a lot and were able to use those very efficiently. With a heavy early focus on architecture, we quickly became familiarized with Spring, javascript, css and Java overall and was able to learn simple ways to use these tools to complete each small test project to know how to do things. While this did not make everything easy, it did allow us to get a good headstart when actually designing the finished web-app.

Another great element of OOAD design was utilizing design patterns within the code. While the Spring MVC was a pattern that came with the territory, other patterns such as Strategy and Factory allowed the code to function much better. We were able to use a factory pattern when creating new sessions based on a game genre. This streamlined things a lot and it made the design much easier to work with. There are also a lot more patterns that could have made the design better, that we can always look towards implementing in the future.

One issue or setback we've experienced is trying to learn new frameworks and tools to accomplish certain things, and sometimes the tools do not come with well made documentation, and is a matter of trial and error, digging through the documentation, or searching through forums of people with the same issue. This led to a member of the team being stuck on a certain feature or aspect of the project that might have a simpler solution. However, this was a great learning experience for everyone in our group and made good progress in learning the foundations for a good developer team and developer life.