Mancala game:

Team Member Names:
Varun Manjunath
Abhinav Gupta
Alexa Rinard

The Final state of the system:

Features implemented:
The features that we implemented in the system are:
Utlilized javascript as the frontend for the system.Used Spring as the backend and with Java code written in the GET and POST call API of Spring.MongoDB database was used to store the state of the system. CSS was used for styling the Frontend of the application. The frontend interface of the system had a screen with two players displayed and the user has to enter the player names. The user clicks on the play button to start the game. The game begins with a board displayed along with the two players on either side of the board. This board was implemented in CSS for styling and Javascript for Frontend. Compared to projects 5 and 6, project 7 was more about integrating the Front end with the backend. An additional MongoDB database was added to store the state of the game like the number of beads(stones) in each cup and mancala. So initially the MongoDB database is initialized with the initial board state. When the user clicks on a cup, the Spring API is called and the Java code is processed to get the next board state, which then updates the MongoDB database. The Front end fetches this updated board state from the MongoDB database. So MongoDB added here acts as a repository of information and aids in the communication between the frontend and the backend code. We could not implement the feature where a human player and a computer player play with each other but rather ended up implementing two human players that play with each other due to a lack of time and the complexity of designing an Artificial Intelligence system for the computer player to play. Certain validation checks like checking if the number of beads in all the cups adds up to the same value couldn't be performed due to lack of time. Also in projects 5 and 6, we assumed that the player needs to move counter-clockwise and deposit the beads into the cups but in Project 7 it was assumed to be clockwise.

Class Diagram:

Thorough UML Class Diagram:

Project 7 class diagram:

https://lucid.app/lucidchart/091900d6-3271-4c98-946a-60a55d692bee/edit?invitationId=inv_6a435c53-6ee3-4dcd-a751-d78168fd8500

Changes from Project 5 class diagram to the final system:

Project 5 class diagram:

https://lucid.app/lucidchart/661434c0-fc86-437c-9e44-c7ec23fbcfc9/edit?invitationId=inv_4ef43aee-3ad9-40f7-ae6a-1c705709a94b&page=0_0#

- In the Project 5 diagram the MVC pattern(illustrated in the diagram for frontend integration changed to utilizing a three-tier architecture where the frontend is javascript, the backend is Spring and the Database is MongoDB as opposed to using Java Swing as the frontend with MVC pattern in Project 5.
- Entrypoint class SpringBootTutorialApplication was added as an entry point to the class. Additional classes were added like index.html to render the view, using DemoController class in Project 7 to make posts and get calls to Java classes.
- Java classes process and store the data in-game repository which extends the mongo repository. This implied that any data stored in-game repository can be viewed in the mongo database.

Project 6 class diagram:

https://lucid.app/lucidchart/ea88e036-8cc9-4052-bc83-e9e68d3040d1/edit?invitationId=inv_4e2bd4c5-7887-47da-ad98-c17160c19356

Key changes from projects 5 and 6

- Utilized Javascript as the frontend of the application instead of Java Swing
- Additionally added a MongoDB database rather than in-memory storage to keep track of the board state while playing the game.
- Added a login page for the game so that players could enter their names(only two players for Mancala Game)

Third-Party code vs. Original code Statement:

Didn’t use third party code

Third-party elements(URL):

- https://www.jetbrains.com/help/idea/spring-boot.html
- https://www.jetbrains.com/help/idea/spring-boot.html

Statement on the OOAD process for your overall Semester Project:

- Faced issues with integrating the Frontend with the back end in the sense that the board was getting updated slowly when a player clicks on a cup on his side. This was because the MongoDB database and the Front end were not in sync
- Implemented a three-tier architecture to work in our code where we have the frontend, backend, and a database that communicate with each other via REST-API calls
- During the analysis phase, faced difficulties in understanding which architecture and frameworks to use but in the end, ended up figuring it out.