CS 319

Object-Oriented Software Engineering

Final Report

RISK

Group 1-G

Adeem Adil Khatri	21801174
Abdulmalak Albeik	21801277
Mannan Abdul	21801066
Maryam Shahid	21801344
Osama Tanveer	21801147
Yahya Mahmoud Ahmed Elnouby Mohamed	21801332

Contents

1. Introduction

Risk is an online multiplayer game that we have implemented with React.js and Redux for the frontend and google firebase for data management at the backend. A GitHub repository was used for code management among the group. [1] It is a multiplayer game as the game logic, servers and UI components were implemented to allow players to play together on a single device. Multiplayer functionalities work as a console application. The game requires users to create an account via signup or login through an existing one. This allows us to manage data for our save/ load feature that permits the user to load previously played games. In terms of implementation, our game does not support the special case for 2 players where, in addition to the 2 players, there is also a neutral army that serves as a strategic buffer between the players. Our game does however support a regular 2 player game with no neutral army to act as a buffer. This feature was cut due to severe time limitations on the project. In addition to the 2 player special case, another feature that hasn't been cut but may not make it to the final product due to time limitations is the dice roll animation. While we do have dice rolls that decide how player turns are sorted, decide the outcomes of a battle (by comparing the highest pairs of dice rolled) etc., they are done in places where they are not visible to the user. We have implemented a dice roll animation component for use, but may not have enough time to actually refactor it to replace the dice simulating the implementation we currently have.

2. Design Changes

2.1 UI Changes

After designing mockups for our Game UI, we decided to keep it as simple as possible and therefore went with a very minimal look. This is to increase the user's experience and provide an easy to play game. All screens consist of only the essential buttons to allow maximum space for showing the game map without clogging it up.

2.2 Changes in Object Model

The changes made to Object Model, worthy of note are as follows:

- We had a user class that was used to store the information of the users registered in the game along with their saved game data. We chose to remove this class in favor of a remote backend that uses Firebase to implement these features. That is, it will authenticate and store users, and the data for all the users that are already registered.
- Similarly, most of the functionality of the SignUp and SignIn class has been implemented using Firebase too.
- Our Game class in the Object Model has also been changed to a class named Board, this class is where all the main game logic is implemented. It can be thought of as the main controller class.
- Our Object Model now includes a class named, GameSetup. This class is where the setup for the game is handled, i.e. no of players selected, player names and colors picked, players sorted.
- Player sorting is done by rolling a die for every player and sorting from highest to lowest, with the highest getting the first turn. In case of a tie, the player with the name entered first is given precedence. This sorting model is different from the actual board game where the player to roll the highest die goes first and the turn moves to their left.
- We have also removed the Dice class from the Object model and the Player class simulates dice rolls when needed. We do have a component that animates dice rolls but its integration may or may not be completed.

- The class named TroopsDeployer is changed to TroopsGiven.
- The army types have been reduced from infantry, cavalry and canon to just infantry. This is because, in the physical board game, these types are used as placeholders to avoid crowding the board, we have no such limitations in a digital board game as we use numbers to denote armies in a given country. The types are still maintained for use in the trade cards however.
- The component named diceRoll has been created to show animated dice rolls. It has been created using the Font Awesome Library of icons and svgs.

3. Lessons Learnt

3.1 Documentation

The quality and thoroughness of analysis and design reports helped us kickstart the implementation process which made us realize how crucial they were for our project. Although we changed multiple classes, our original class diagram acted as a skeleton for our work. Moreover, many classes were relatively easier to create using the dynamic models we had created for them such as trading cards, processing a player's turn with respect to other players. Modelling code to reflect how our diagrams modeled how the game would work proved to be very helpful acting as a path to tread on.

3.2 Design

We introduced multiple design patterns during implementation. This consisted of strategy pattern, singleton pattern and model-view-controller pattern. These design patterns have been modelled in a class object model but have not been reflected in our code. This is because while we are aware of how design patterns would help simplify code, we also have to acknowledge we are not very familiar with them and were spending too much time trying to implement them. We therefore decided to code our game logic first and then refactor our code to reflect our design pattern object model.

As much as we focused on the game logic, we also realized the importance of giving the

user a better experience. We learnt this once we were testing each other's' codes. Although the person had implemented it correctly, it was difficult for the tester to navigate through the work. We implemented alerts for validation and worked on a readable and easy to navigate UI.

3.3 Communication

Initially, we held meetings after every 14 days. This gave us enough time to plan and brainstorm our project. The meetings would only end once everyone was on board with the plan. During the implementation process, we met daily for 30 minutes. Since 6 people were developing a single game in a limited time and remotely, we found it difficult to manage and resolve conflicts between our code. This is why we added daily meetings as they allowed everyone to be in sync and show what they were working on.

3.4 Testing

We delayed testing towards the end which added pressure to resolving problems that came up during the testing process. However, we soon learnt the importance of testing and started running and checking each other's codes as soon as it was pushed to the repository. This allowed us to catch mistakes and errors before it was too late and difficult to manage them.

4. User’s Guide

4.1 Introduction

Risk is an online multiplayer game for 3-5 people with a simple goal - conquering the map. This involves capturing your enemy’s territories by deploying troops and engaging in battle. Depending on the roll of your dice, you either conquer land or lose it. Defeating your enemy’s troops allows you to move forward on your path of global domination.

Objective of the Game: Conquer the map by occupying all territories.

4.2 System Requirements

Since we are running the game in a browser, our minimum requirements are those of our chosen browser, which is chrome. They are as follows:

- Windows 7, Windows 8, Windows 8.1, Windows 10 or later
- An Intel Pentium 4 processor or later that's SSE3 capable
- 2 GB RAM or more
- Keyboard and mouse
- Operating System: Windows or MacOS
- Note: Javascript should be enabled in Chrome

4.3 How to play

4.3.1 Login Screen

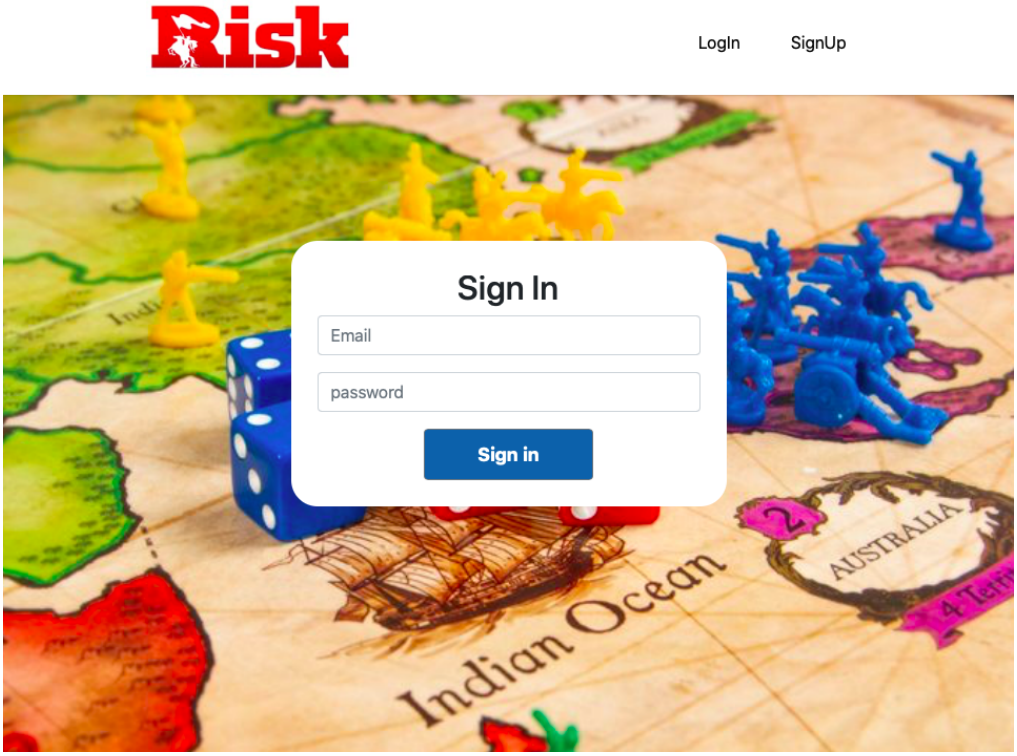

Figure 1

- The user is required to sign in before beginning the game.
- Sign in consists of an email and password for verification.
- In case the user does not have an account, he/she can choose to sign up using the sign up button.

4.3.2 Signup Screen

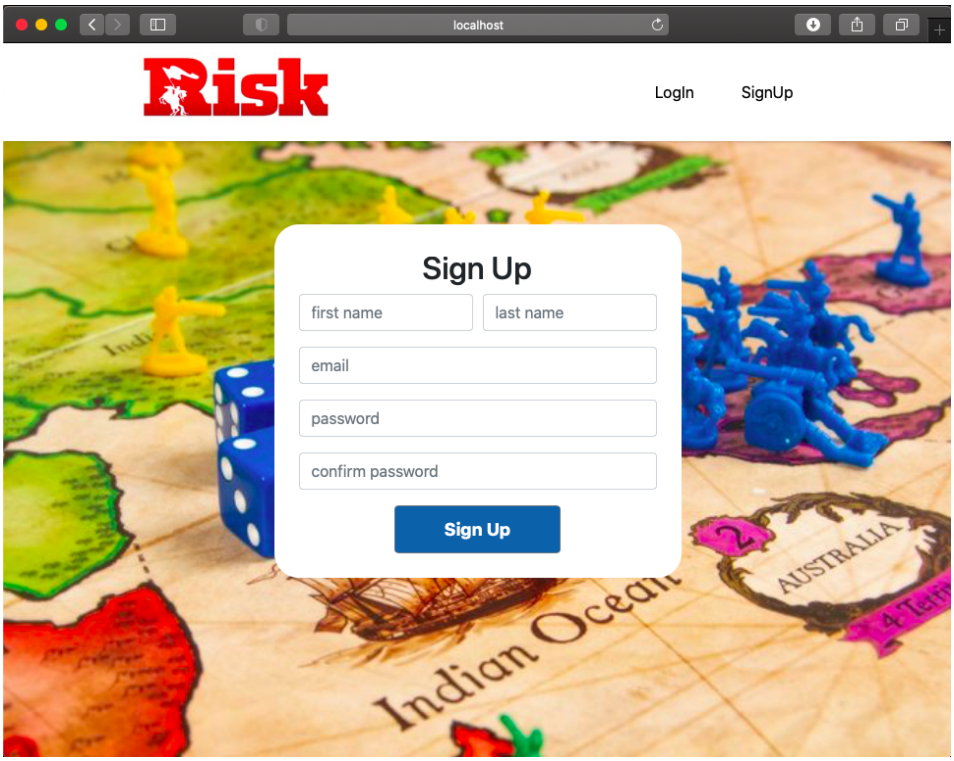

Figure 2

- Sign up screen requires the user to enter their first and last name for identification.
- It also requires an email and password that will be used by the user to log into the game.
- The user's information is stored once he/ she clicks the sign up button.

4.3.3 Load Game Screen

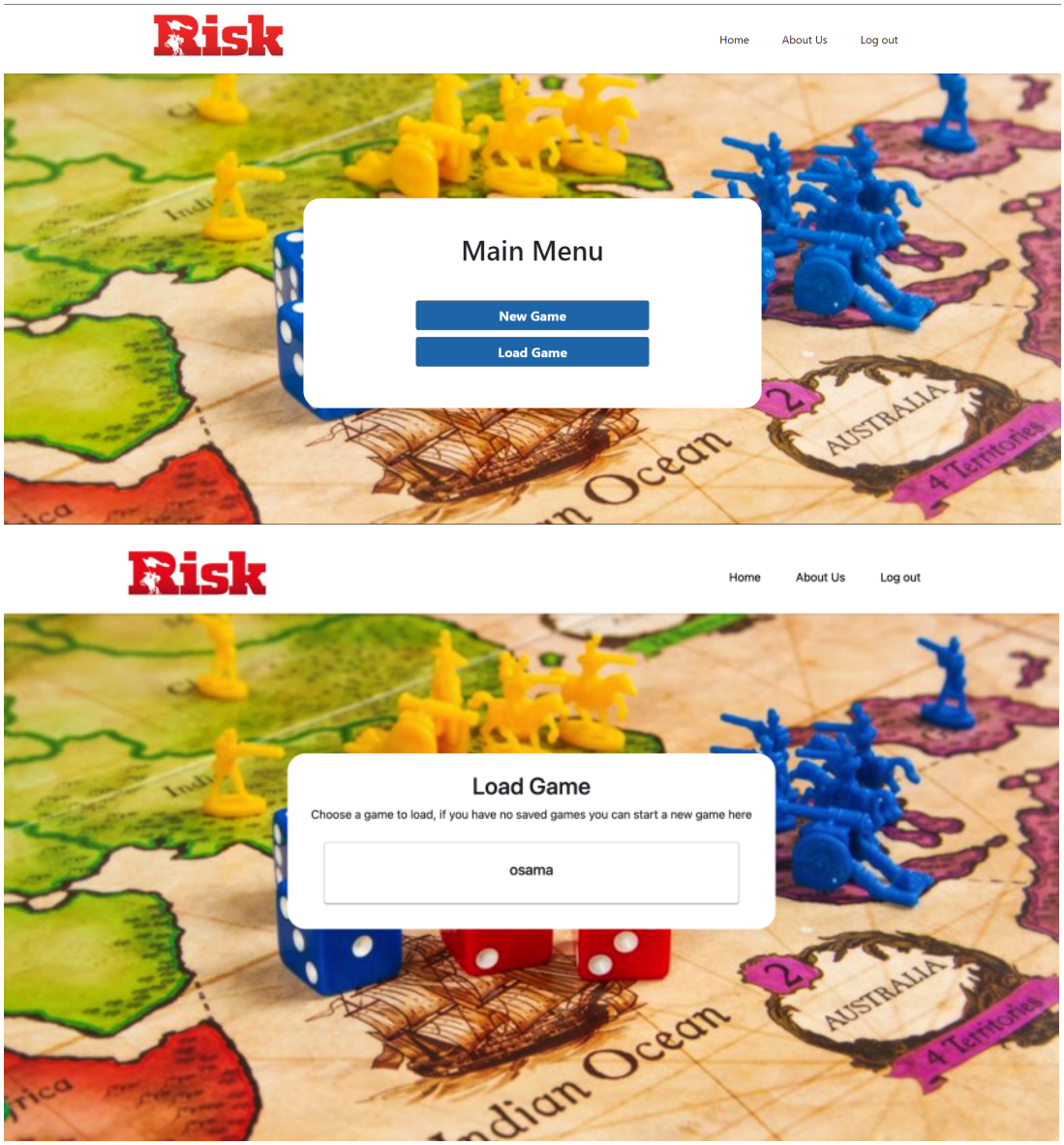

Figure 3

- The user can choose to load a previously saved game using the load game function. This is accessed from using the load game option that is presented after sign-in that shows 2 options, new game, load game in the main menu.

4.3.4 Game Setup Screen - Number of Players

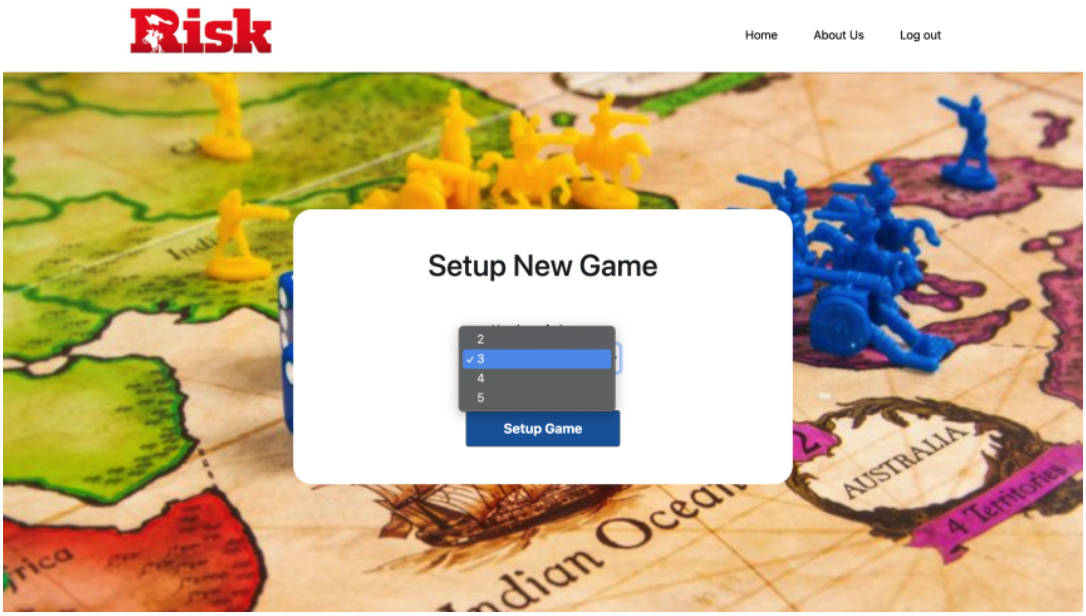

Figure 4

- Set-up new game screen consists of two parts. The first part asks the user to choose the total number of players between 2 to 5.
- The number of players is finalized by pressing the Setup Game button.

4.3.5 Game Setup Screen - Enter Player Name and Color

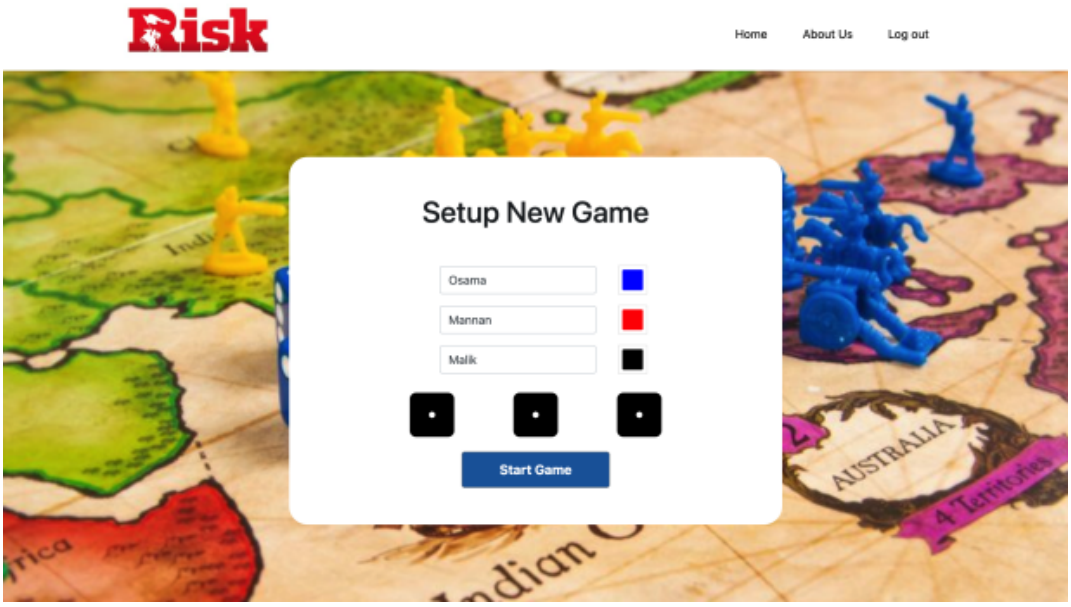

Figure 5

- The second screen for setup consists of the name of players and their respective colour.
- The colour can be chosen from a variety of options.
- When we click start game, the dice are rolled to determine the player turns and their animations are shown below.

4.3.6 Game Screen - Player Turn

Figure 6

- The gameplay holds the map in the centre which consists of the number of troops deployed by the players - represented by their respective colour.
- The player with the current turn will be highlighted in golden. He/she can choose to attack, fortify or pass.
- To deploy troops on a valid territory, the player needs to double click on it.
- To attack a territory, the player needs to click on the territory they want to attack from and then click on the territory they want to attack. 2 text-boxes appear, the first taking the number of dice the attacker wants to roll and the second the number of dice that the defender wants to roll.
- A player's trade cards can be accessed from the icon showing the cards on the left of the screen.

4.3.7 Player Information

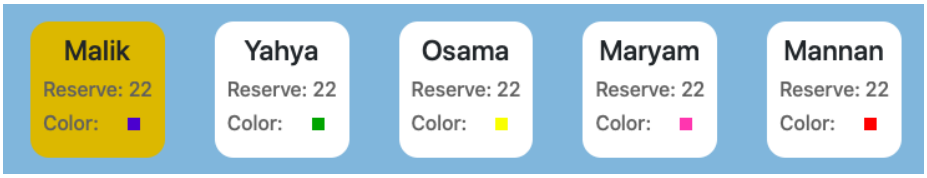

Figure 7

- The top of the screen shows the name, colour and number of troops of each player.
- The player with the current turn is highlighted in golden.

4.3.8 Trade Cards

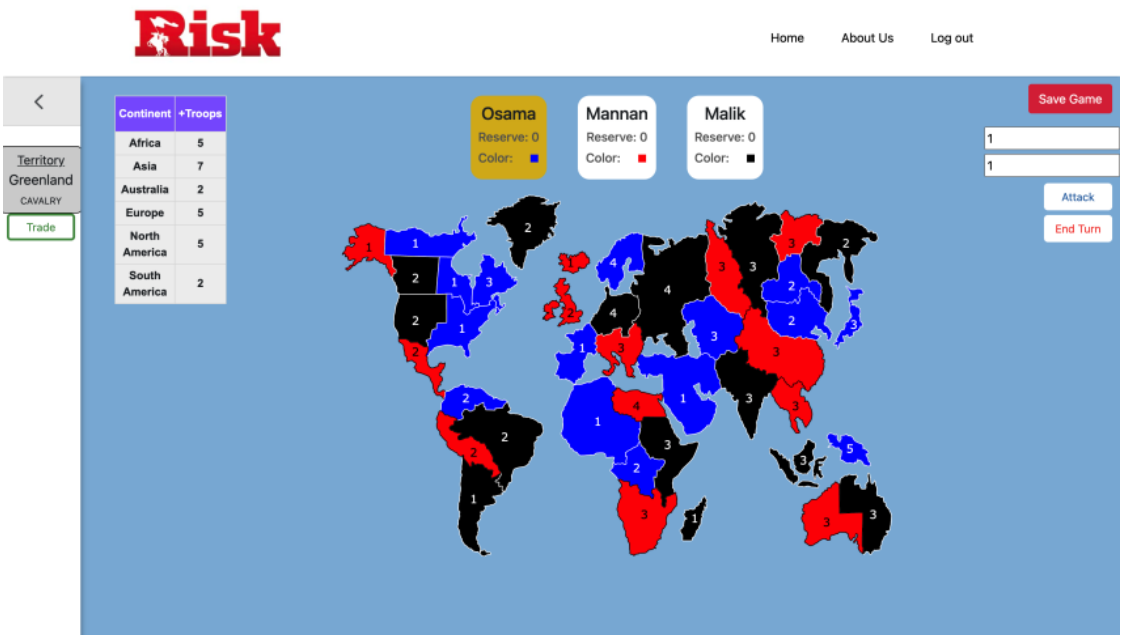

Figure 8

- After a player clicks on the trade card icon, the above menu will open which will show all the trade cards a player has.
- To trade in a set of cards, the player clicks on 3 cards to select them and then clicks on the trade button to trade them in.

4.3.9 The Map

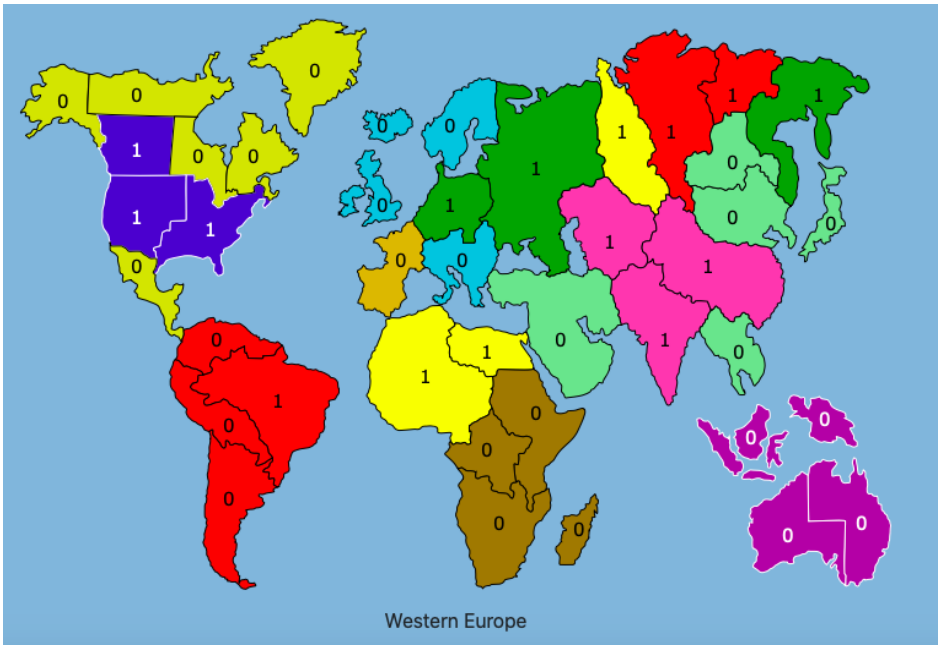

Figure 9

- The map highlights countries of each continent to a specific colour.
- The colours then change according to the colour of the player occupying the territory.
- The name of the selected country is shown at the bottom.

4.3.10 Table of Continents and their Troops

Continent	+Troops
Africa	5
Asia	7
Australia	2
Europe	5
North America	5
South America	2

Figure 10

- This table shows the number of additional troops a player gets each turn if they control all the territories in a continent.

4.3.11 Save Screen

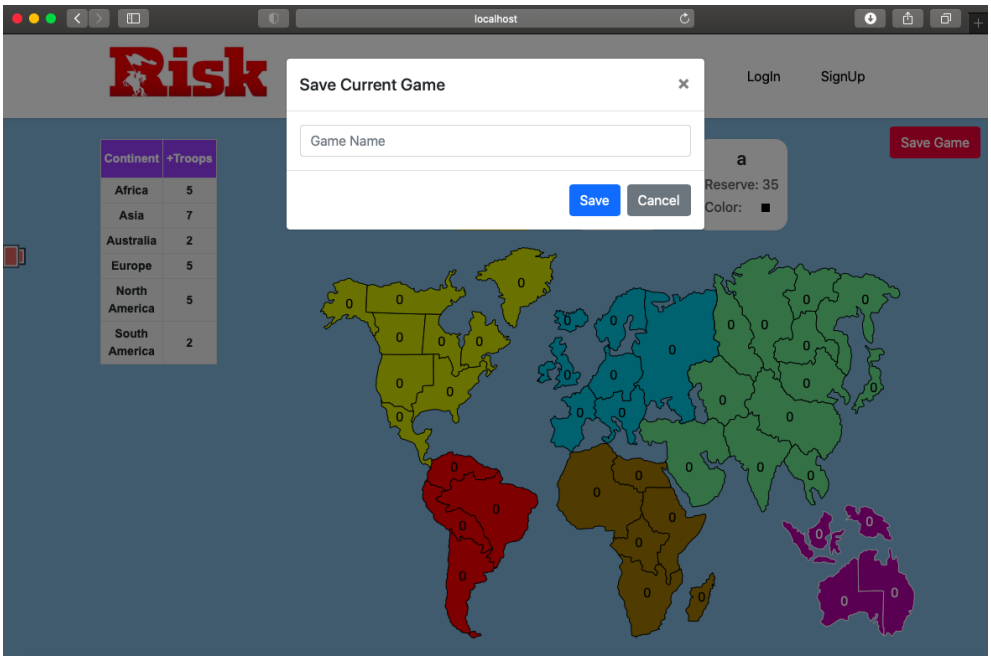

Figure 11

- The player can choose to save their game at any time using the red save game button. The pop up shown then appears where we can enter the name of the save game and proceed to saving it.

5. Build Instructions

1. Install Google Chrome as it is some group members' browser of choice.
2. The game can be accessed using the following link: https://d1gwexiayrw7gx.cloudfront.net/

6. Work Allocation

6.1 Adeem Adil Khatri

- Created the UI mockup designs of the game for the Analysis Report. [1]
- Responsible for SubPackage Decomposition Diagram in Design Report.

- Made the svgs of the Map (1st Iteration)
- Revised the Class/Object Diagram.
- Video Trailer for Demo (In Progress)

6.2 Abdulmalak Albeik

- Backend integrations.
- Signin and Signup implementations.
- Saving multiple games and loading them to play again.
- Redux and cross components data management implementations.
- General UI implementation and improvements.
- Class & Object diagrams.
- CI/CD Pipeline integration.
- AWS Hosting, and caching configs.

6.3 Mannan Abdul

- Use-case diagram and its explanation in the Analysis Report [2].
- Access control and security, Boundary Conditions (Initialization, Termination, Exceptions) parts in the Design Report [3].
- State diagram and its explanation in Analysis design report Iteration 2 [2].
- Implementation of the Dice Roll animation using the Font Awesome Library and importing all needed icons into a local library.
- Integration of Dice Roll animation where needed.
- General contribution to different classes like MapPaths, Map, Board, GameSetup, gameConstants, Die, RollDice and their css etc.
- Game Logic testing according to the risk manual to check functionality of the game..
- Bug fixes

6.4 Maryam Shahid

- Created the UI mockup designs of the game for the Analysis Report. [1]
- Wrote the design goals of the system for the Design Report. [2]
- Created the presentation for the demonstration.
- Was responsible for the second iteration of the Analysis Report according to the feedback given.
- Tested the code of both UI and game logic according to the risk manual to test functionality.
- General contribution to classes.
- Bug fixes.

6.5 Osama Tanveer

- Sequence/Activity/State Diagrams (1st Iteration)
- Object Diagram/Design Trade-Offs (2nd Iteration)
- Placement of SVGs on the map.
- Interactiveness of all game objects
- Implemented Attack/Defense and Maneuver Logic
- Integrated Trade Card system into the game
- Bug fixes.
- Redesigned UI.

6.6 Yahya Mahmoud Ahmed Elnouby Mohamed

- Introduction, Overview, Functional and non-functional requirements in Analysis Report, proofreading the report(1st iteration)
- Hardware/Software mapping and deployment diagram in design report, proofreading the report (1st iteration)
- Revising and fixing the analysis report according to the feedback, adding a new sequence diagram, updating the full turn activity diagram (2nd Iteration)
- Revising Hardware/Software mapping and proofreading, updating the design report.

- Main Menu Implementation and UI
- Login Initial UI design
- Trade System logic implementation
- Updating color picker in setup page
- Bugs fixing

7. References

[1] "malikalbeik/1G-Risk-game", Github ,2020. [Online]. Available: https://github.com/malikalbeik/1G-Risk-game [Accessed: 20- Dec-2020].

[2] "2nd Iteration CS319 Analysis Report", Github , 2020. [Online]. Available: https://github.com/malikalbeik/1G-Risk-game/blob/master/2nd%20Iteration%20CS319%20Analysis%20Report.pdf [Accessed: 20- Dec-2020].

[3] “2nd Iteration CS319 Design Report” Github, 2020. [Online]. Available:

https://github.com/malikalbeik/1G-Risk-game/blob/master/2nd%20Iteration%20CS319%20Design%20Report.pdf [Accessed: 20-Dec-2020].