Beer Game Proposal

Software Engineering

Prof. Dr. Peter Bauman

Mahiem Agrawal
Arnav Singh
Ishwor Giri
Subigya Paudel
Maulik Chhetri

Contents

1 Introduction

1.1 General Idea

The beer game is based on supply chain and involves 4 players where each player represents one of the four stages in the chain i.e the retailer, the manufacturer, the wholesaler, the consumer.

Every week the retailer receives orders from the consumer without any time delay. The retailer fulfills as many unit of beer order as it can and then calculates the number of units that are required based on the total number of stocks left in the inventory, the cost of each beer, and the backlog.

After some time delay, the retailer places an order to the wholesaler. The wholesaler tries to fulfill the demand of the retailer and estimates how much beer is needed. The wholesaler places an order to the distributor if he senses he is in need of more stocks. This process is repeated with the manufacturer as well.

Lastly, the manufacturer begins production according to the order demand. When the production is over, the beer is send backwards to the distributor, then from the distributor to the wholesaler, retailer, and finally to the consumer. There are time delays in each step during the downstream process, except between the retailer and consumer.

1.2 Goals and Objectives

The objective of the game is to receive orders and deliver the items back to the consumer, in a way the maximizes profits and minimizes the total cumulative cost of all stages based the given information. Cost can be arisen from numerous places. There is a cost for holding the inventory, a cost for not satisfying demand so called a "back order" which is basically a cost that is incurred until the demand for the product is satisfied.

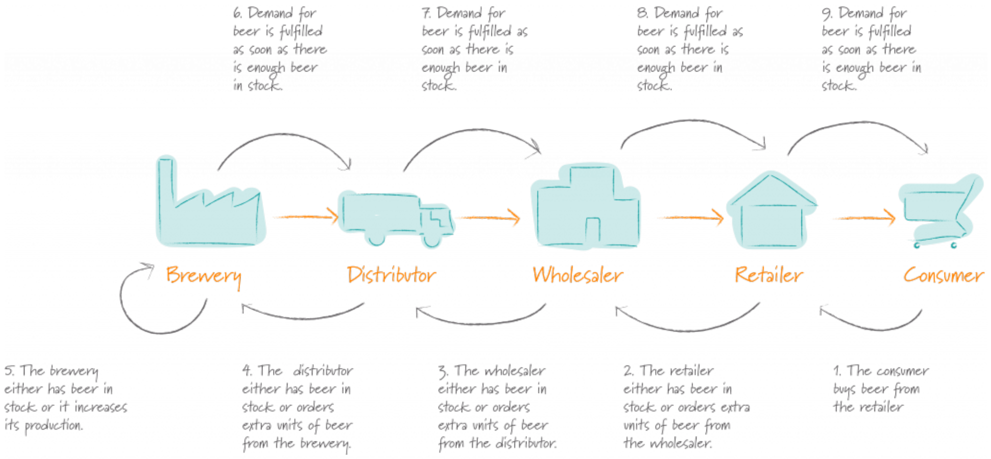
6. Demand for beer is fulfilled as soon as there is enough beer in stock.
7. Demand for beer is fulfilled as soon as there is enough beer in stock.
8. Demand for beer is fulfilled as soon as there is enough beer in stock.
9. Demand for beer is fulfilled as soon as there is enough beer in stock.
Brewery
Distributor
Wholesaler
Retailer
Consumer
5. The brewery either has beer in stock or it increases its production.
4. The distributor either has beer in stock or orders extra units of beer from the brewery.
3. The wholesaler either has beer in stock or orders extra units of beer from the distributor.
2. The retailer either has beer in stock or orders extra units of beer from the wholesaler.
1. The consumer buys beer from the retailer

Figure 1: Diagram to demonstrate the Supply Chain and and the flow of information

2 Requirements

2.1 User

The user can be logged in as a student or as an instructor. Each of these roles have their own requirements.

2.1.1 Instructor

- The instructor acts as as an administrator of the game.
- One should be able to modify the input parameter, record every players individual performance and see the summary of the game results.
- The instructor must be able to set the initial values for the following parameters prior to the start of each game.
  1. Number of rounds, weeks per game
  2. Time delay for both upstream and downstream
  3. Initial number of items in inventory of each stakeholder
  4. The backlog and the cost of each item in stock
  5. The amount of information available to each stakeholder can view in the game

- The instructor must be able to supervise the current state of each game during a session.
- The instructor must be able to change the type of demand for the players.
- The instructor and user must be able to view a plot showing the bullwhip effect and player demand versus demand generated.

2.1.2 Player

Once the lobby and game setting are initialised by the instructor players can join the lobby.

- A player should be able to decide on his/her role be able to choose a role of Retailer, Wholesaler, Distributor, or Factory.
- A player shall be able to place orders to the upstream supply chain.
- The players are not allowed to discuss or talk about their own strategies with each other. Players can only view the following:
  1. The backlog of orders that couldn't be fulfilled.
  2. Incoming orders from the upstream
  3. Number of stocks in the inventory
  4. Updated information after the orders have been placed.

There are 24 rounds of the game and in each round the player performs the following 4 steps.

1. **Check Deliveries** Check the quantity of beer units that are delivered to the downstream supplier in the supply chain within some time delay.
2. **Check Incoming Orders** Check the quantity of beer units that the downstream supplier has ordered within some time delay.
3. **Deliver Beer** To satisfy the consumer demand, deliver as many units as possible.
4. **Place Out going Order** This is the decision-making process where the number of units of beer needed are decided to keep the inventory going and ensure that he has sufficient units to meet future demands. The decision is based on 3 factors namely backlog, consumer needs, and current stock.

2.2 System

Following are the system requirements that must be kept it when creating the game. Please note that there maybe other requirements as well however these are the major ones that need be talked upon and the foundation through which the game should be created.

1. When first opening app user (who could either be a player or an instructor) should be presented with the options to login or to register.

2. A player should only be able to play a game which is has been created by some instructor.

3. There can only be one instructor for one session or game that is happening after which he will receive a unique code only for that session.

4. User can only join the game by typing the unique code after which he should automatically be assigned the role of a player.

5. A player should only be active in one session.

6. All players and instructor should have their own unique id to identify them.

7. When the game reaches the maximum number of weeks, it should terminate and present players with a final interface which displays the saved game statistics.

8. Players should be able to choose a valid role (Factory, Distributor, Wholesaler and Retailer) that they must stick to and cannot be changed.

9. After each round the system should be able to show the player the demand from the successor, the incoming beers from the predecessor, the number of beers in warehouse, the back log and input field to place the order.

10. Certain settings such as holding costs or beer costs cannot be changed by the instructor once the game has started but certain settings such as number of rounds should be able to change once the game has instructed. The system should be able to handle such functionalities.

11. The instructor should be able to inspect each game in the session while it is ongoing and after the game has ended the instructor should be able to see weekly analysis after each round.

12. When the game reaches the maximum number of weeks, it should terminate and present players with a final interface which displays the saved game statistics.

3 Design

3.1 Use Case

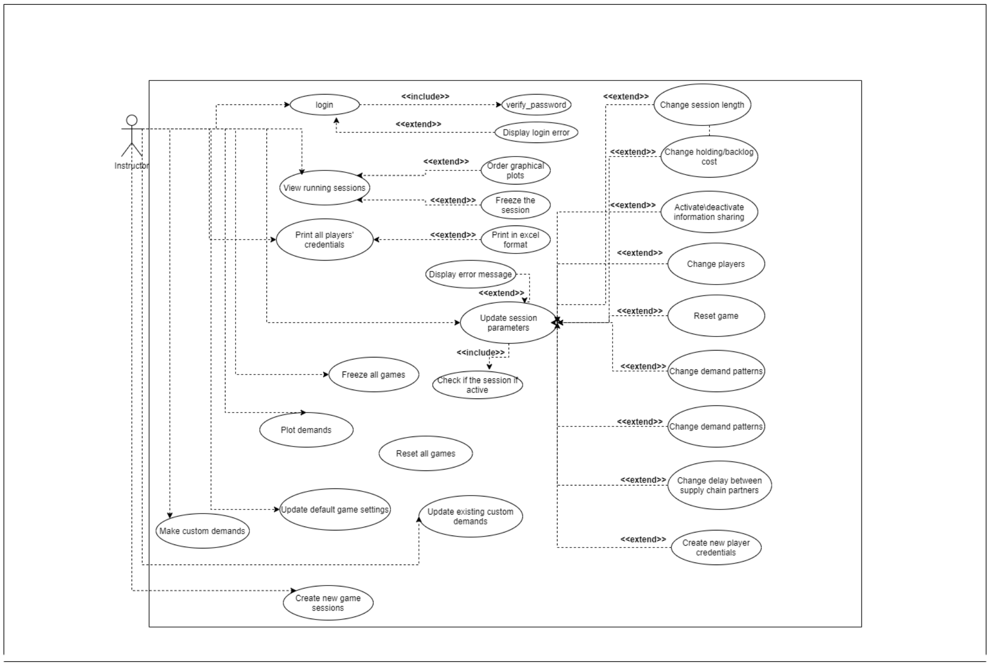

Figure 2: Use case diagram for the instructor

The above use-case diagram provides a comprehensive overview of what an instructor can do using the web application, from logging in the the game to changing the properties of all the games that instructor is managing. The diagram also captures the requirement of the software to edit the various game parameters of a particular game that s/he is managing.

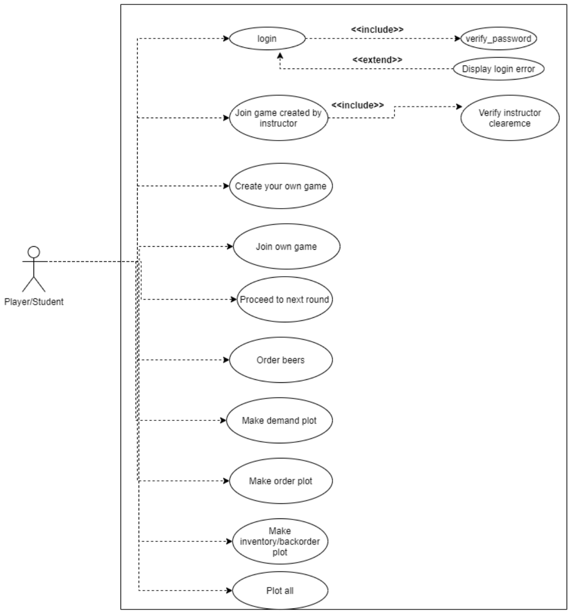

Figure 3: Use case diagram for the player

The above use-case diagram gives an overview of the functionalities that the web service offers to the players of the beer game. It includes every action that the user can take while playing the game, like ordering the beers and making plots of the back order, demand or/and total cost. It also encompasses use cases that are not necessarily involved with playing the game, like creating one's own game.

3.2 Class

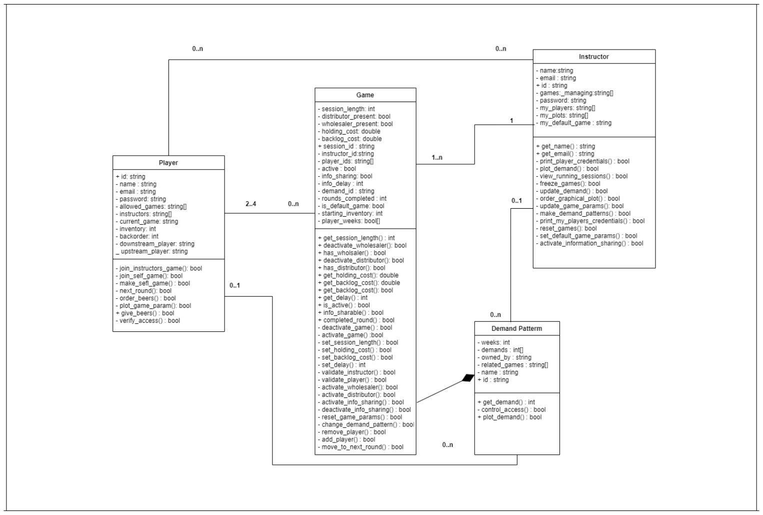
0..n
0..n
Instructor
- name:string
- email : string
+ id : string
- games:_managing:string[]
- password: string
- my_players: string[]
- my_plots: string[]
- my_default_game : string
+ get_name() : string
+ get_email() : string
- print_player_credentials() : bool
- plot_demand() : bool
- view_running_sessions() : bool
- freeze_games(): bool
- update_demand() : bool
- order_graphical_plot() : bool
- update_game_params(): bool
- make_demand_patterns() : bool
- print_my_players_credentials() : bool
- reset_games(): bool
- set_default_game_params() : bool
- activate_information_sharing() : bool
Game
- session_length: int
- distributor_present: bool
- wholesaler_present: bool
- holding_cost: double
- backlog_cost: double
+ session_id : string
- instructor_id:string
- player_ids: string[]
- active : bool
- info_sharing: bool
- info_delay : int
- demand_id : string
- rounds_completed : int
- is_default_game: bool
- starting_inventory: int
- player_weeks: bool[]
+ get_session_length() : int
+ deactivate_wholesaler(): bool
+ has_wholsaler() : bool
+ deactivate_distributor(): bool
+ has_distributor(): bool
+ get_holding_cost(): double
+ get_backlog_cost(): double
+ get_backlog_cost(): bool
+ get_delay() : int
+ is_active() : bool
+ info_sharable() : bool
+ completed_round() : bool
- deactivate_game() : bool
- activate_game() :bool
- set_session_length() : bool
- set_holding_cost() : bool
- set_backlog_cost() : bool
- set_delay() : int
- validate_instructor() : bool
- validate_player() : bool
- activate_wholesaler(): bool
- activate_distributor(): bool
- activate_info_sharing() : bool
- deactivate_info_sharing() : bool
- reset_game_params() : bool
- change_demand_pattern() : bool
- remove_player() : bool
- add_player() : bool
- move_to_next_round() : bool
1..n
1
Player
+ id: string
- name : string
- email : string
- password: string
- allowed_games: string[]
- instructors: string[]
- current_game: string
- inventory: int
- backorder: int
- downstream_player: string
_ upstream_player: string
- join_instructors_game(): bool
- join_self_game(): bool
- make_sefl_game() : bool
- next_round(): bool
- order_beers() : bool
- plot_game_param(): bool
+ give_beers() : bool
- verify_access() : bool
2..4
0..n
0..1
0..1
0..n
Demand Patterm
- weeks: int
- demands : int[]
- owned_by : string
- related_games : string[]
- name : string
+ id : string
+ get_demand() : int
- control_access() : bool
+ plot_demand() : bool
0..n

Figure 4: Class diagram for the implementation

3.3 Activity

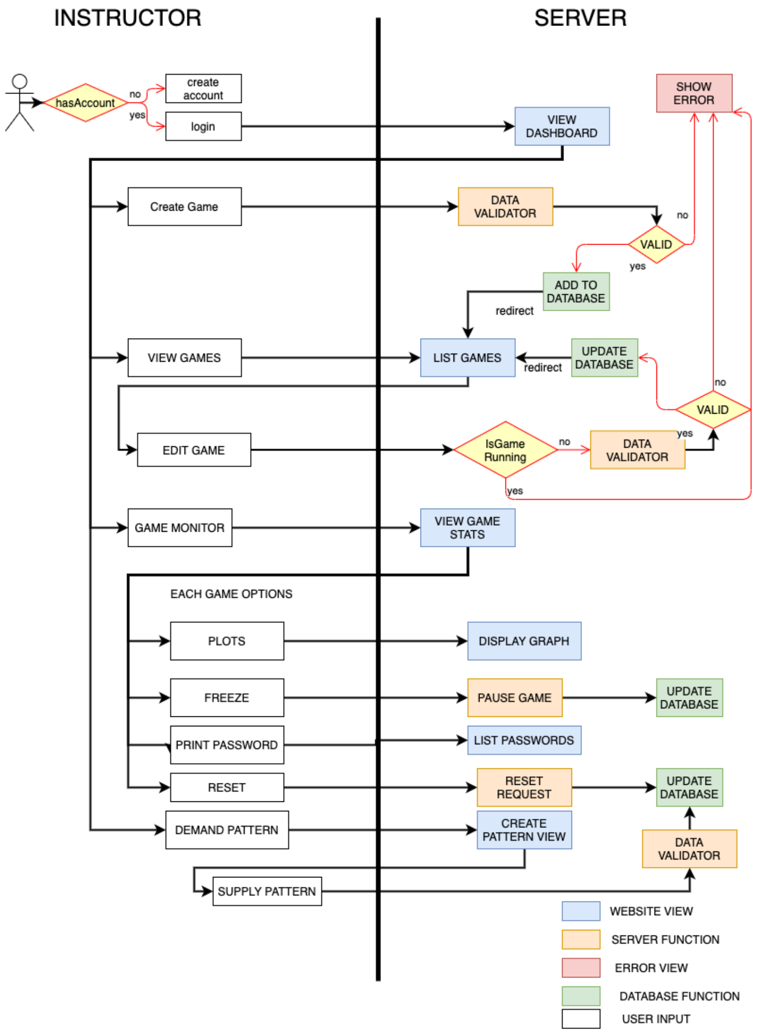

Figure 5: Instructor Activity View

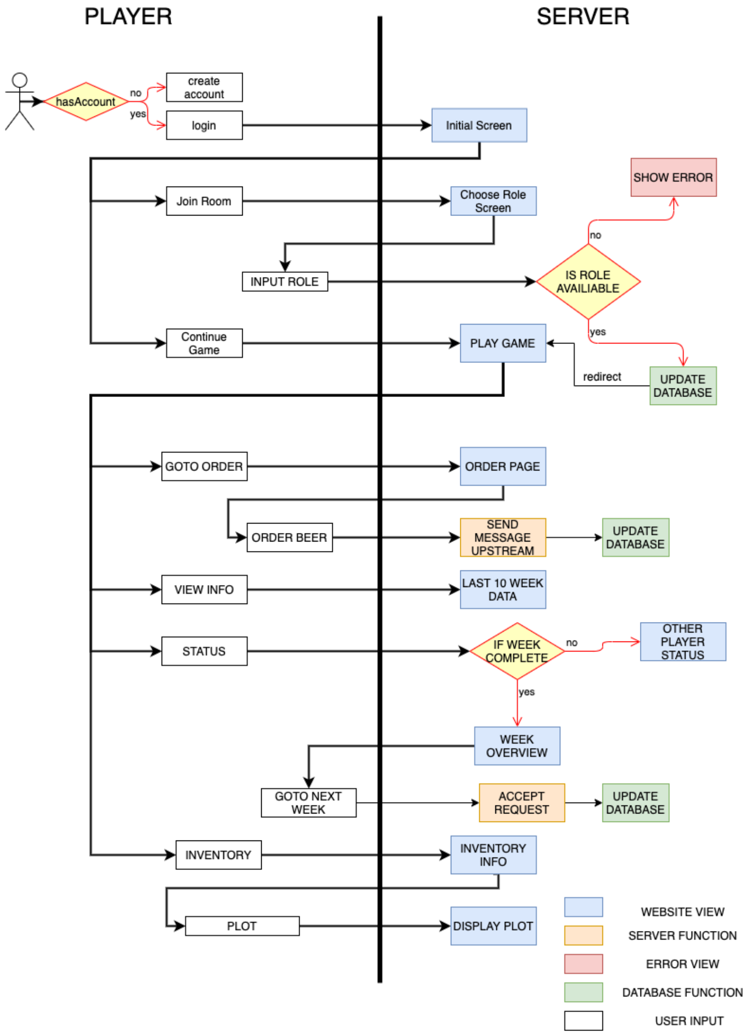

Figure 6: Player Activity View

3.4 Sequence

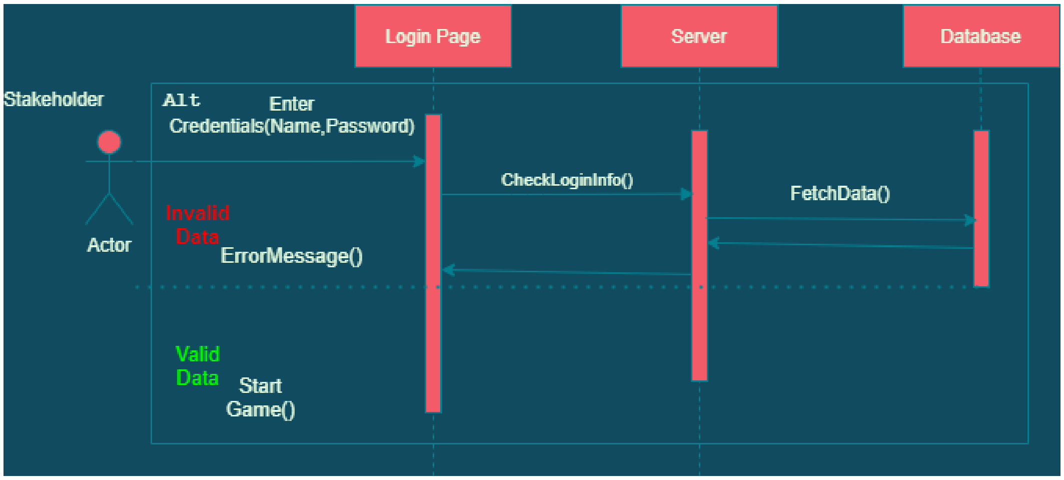

Figure 7: Sequence Diagram for Login Page for stakeholders

The Stakeholder tries to login to the game with the relevant credentials-name and password, and allowed to interact with the game only if he/she is registered for the game.

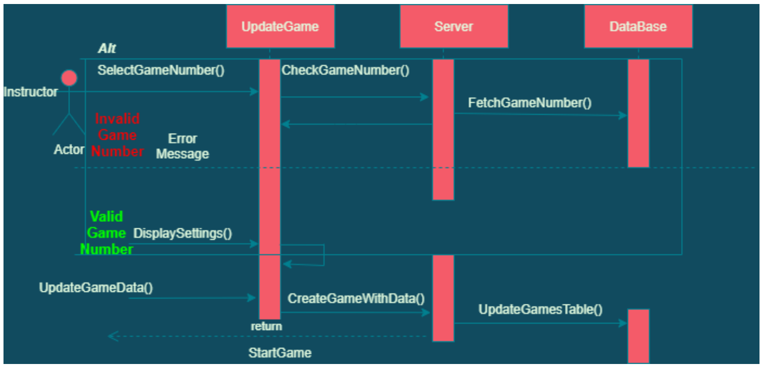

Figure 8: Use case diagram for the player

The following sequence diagram explains how an instructor can update a game that has been created. The instructor can change the relevant settings of the game, for example increasing or decreasing the backlog cost.

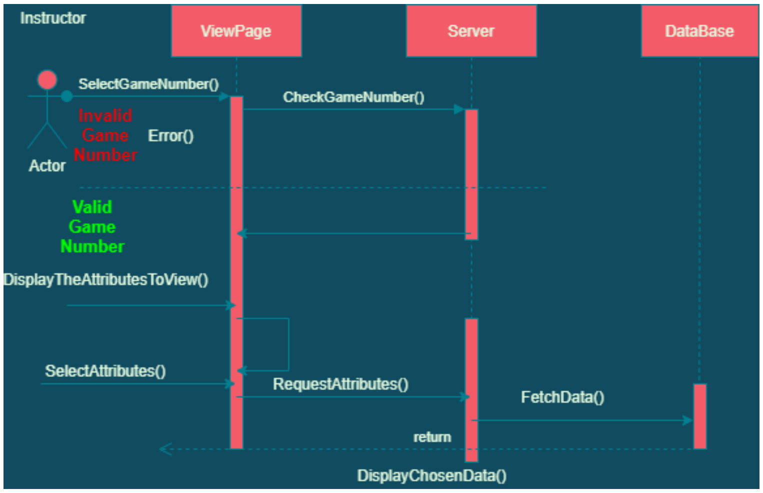

Figure 9: Sequence Diagram for viewing a game for instructor

This sequence diagram explains how the instructor is able to view the data of a particular game.

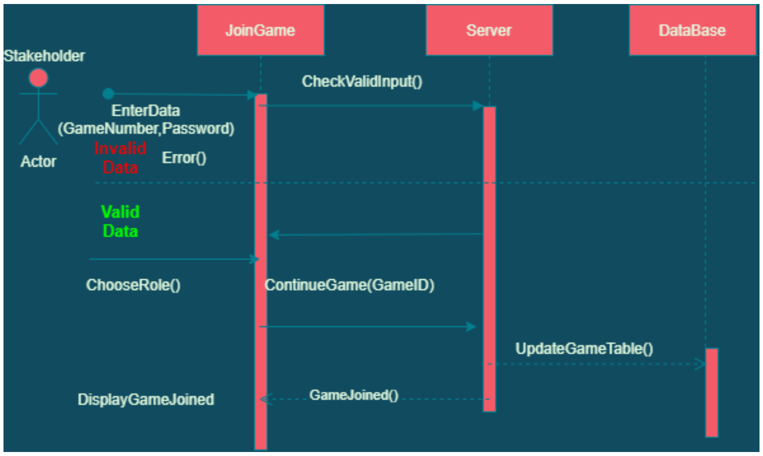

Figure 10: Sequence Diagram for joining a game by a player

The following sequence diagram explains how a player can join a game. The player requires a game number that is registered, password, and the role in which the player tends to join. The server then checks for the validity on the basis of given information and responds whether or not the player has been able to join the game.

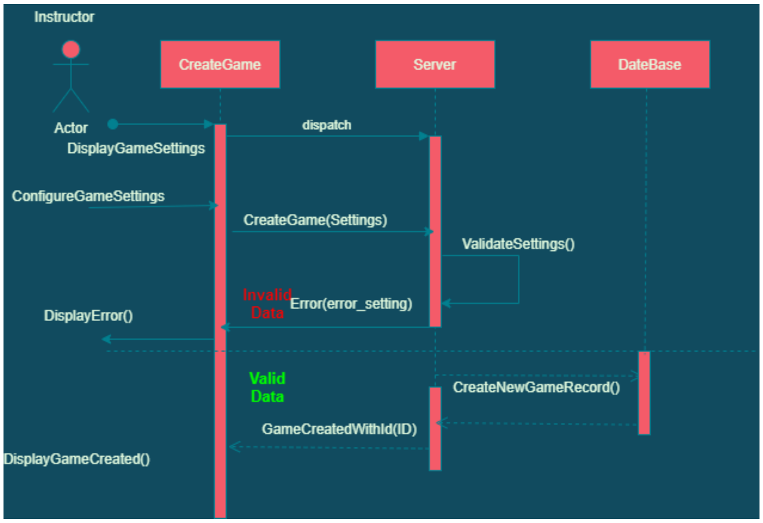

Figure 11: Sequence Diagram for creating a game by a instructor

The following sequence diagram explains how the instructor can create a game.The instructor has to configure the settings of the game that would be displayed. Then, by providing a valid number(which also is a configuration step), the instructor will be able to create a new game.

4 Conclusion

The goal of this game is to teach students who use this software about **The Bullwhip Effect** which is a supply chain phenomenon describing how small fluctuations in demand at the retail level can cause progressively larger fluctuations in demand at the wholesale, distributor, manufacturer and raw material supplier levels. The effect is named after the physics involved in cracking a whip. When the person holding the whip snaps their wrist, the relatively small movement causes the whip's wave patterns to increasingly amplify in a chain reaction.

The purpose of this document is for developers to have a guideline to follow and know procedures about how to do your coding instead of just starting form scratch. The ideas presented above teach you with general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code.

5 References

1. https://www.transentis.com/understanding-the-beer-game/en/
2. http://scgames.bauer.uh.edu
3. https://www.draw.io/
4. https://en.wikipedia.org/wiki/Beer distribution game
5. https://www.supplychain-academy.net/beer-game/
6. http://www.peter-baumann.org