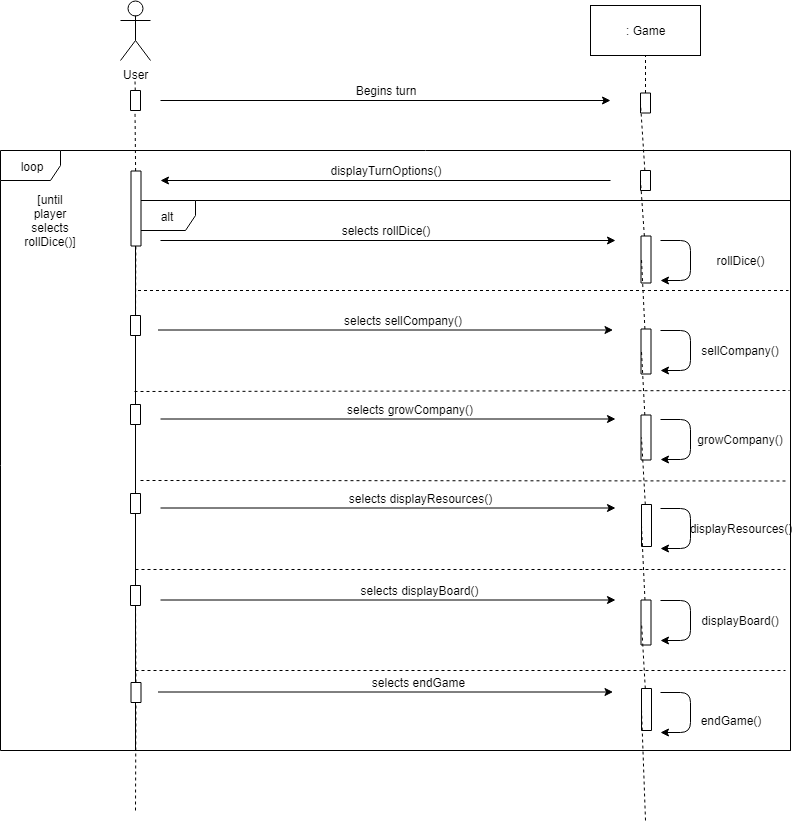
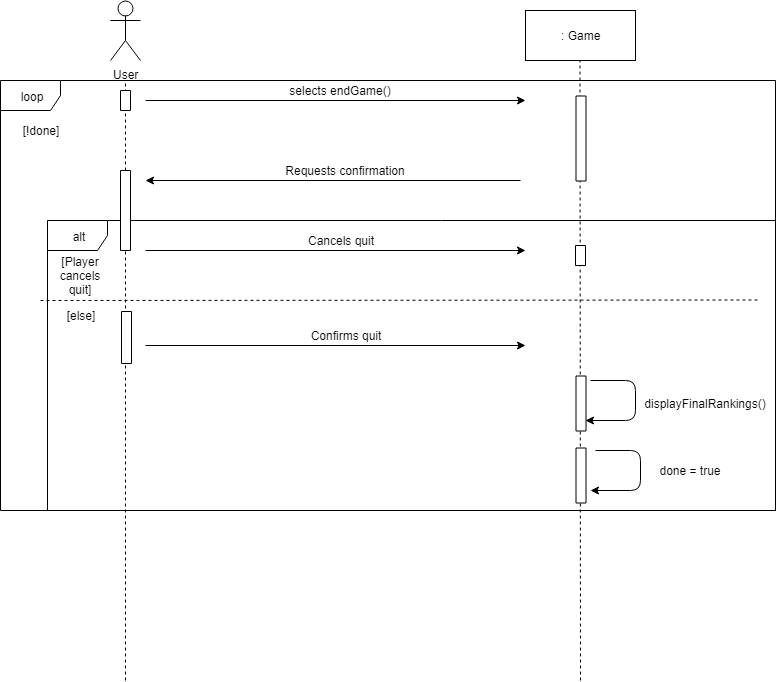


1. **Select Number of Players.** The system loops through a series of console outputs, asking for the number of players and then confirmation of the number of players. If the user has entered the incorrect number of players, they can indicate this and the loop will restart. Only when the user inputs confirmation of the number of player will the loop break and requestPlayerNames() be called.

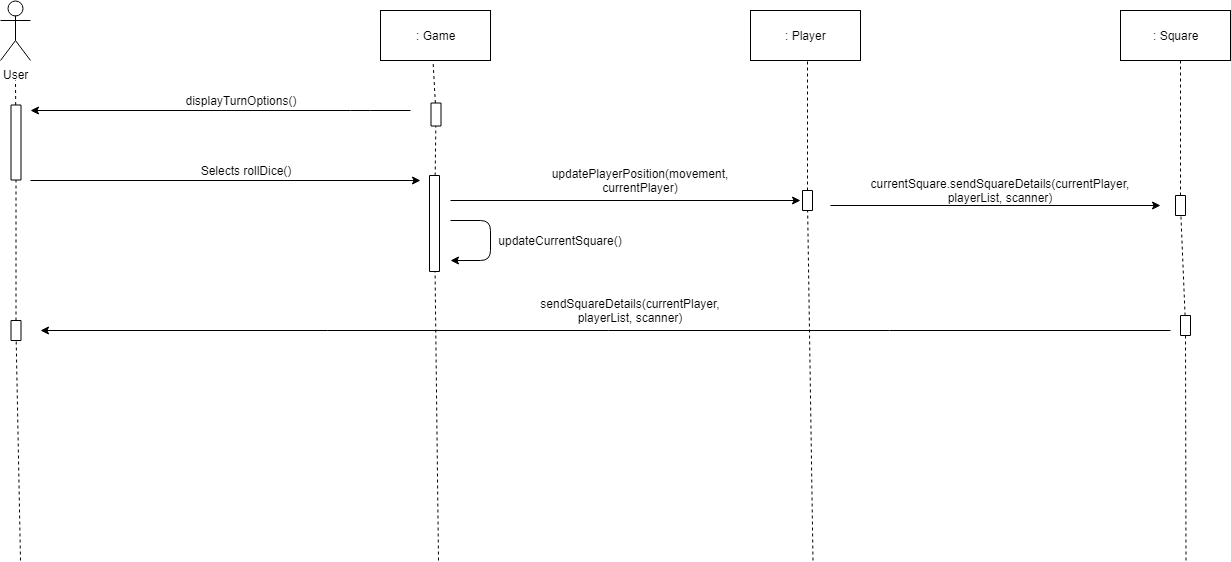
**2. Request Player Names.** The system loops through a series of console outputs asking for the players’ names. The outer loop is executed as many times as there are players and the inner loop breaks only when each name is confirmed. If the player inputs an existing name or fails to confirm their input, they are placed back at the beginning of the inner loop.

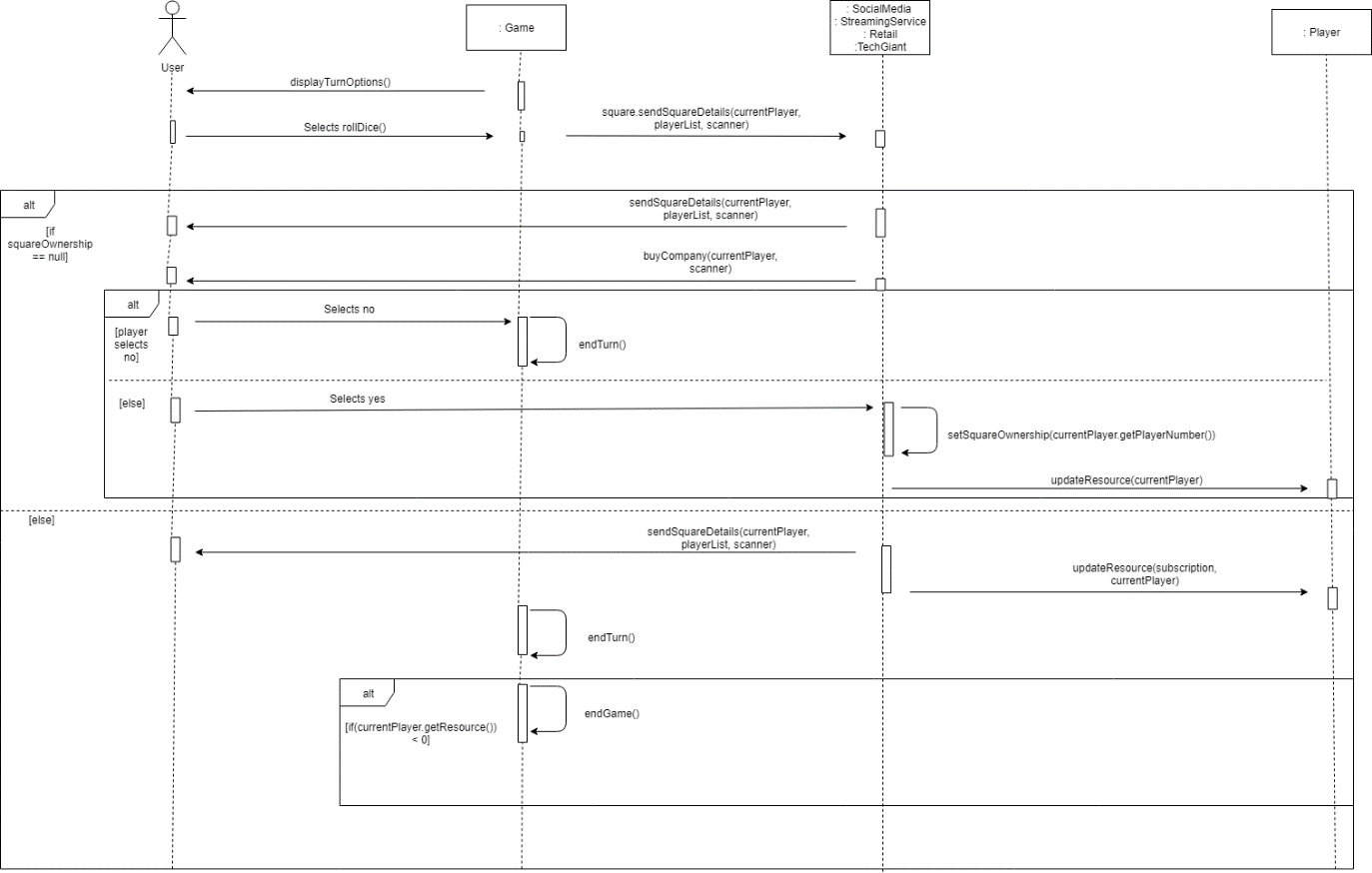
**3. Display Turn Options.** The system displays the turn options to the user. Each of these options, when selected, will result in a method call. The menu loops until the player selects rollDice().

**4. Quit Game.** If the player selects endGame() from the displayTurnOptions() menu, the system will request confirmation via a system output.

If the player inputs “N”, the quit will be cancelled. If the player inputs “Y”, then the quit will be confirmed. displayFinalRankings() is called to show the final scores of the players and done is set to true to terminate the endGame() menu loop.



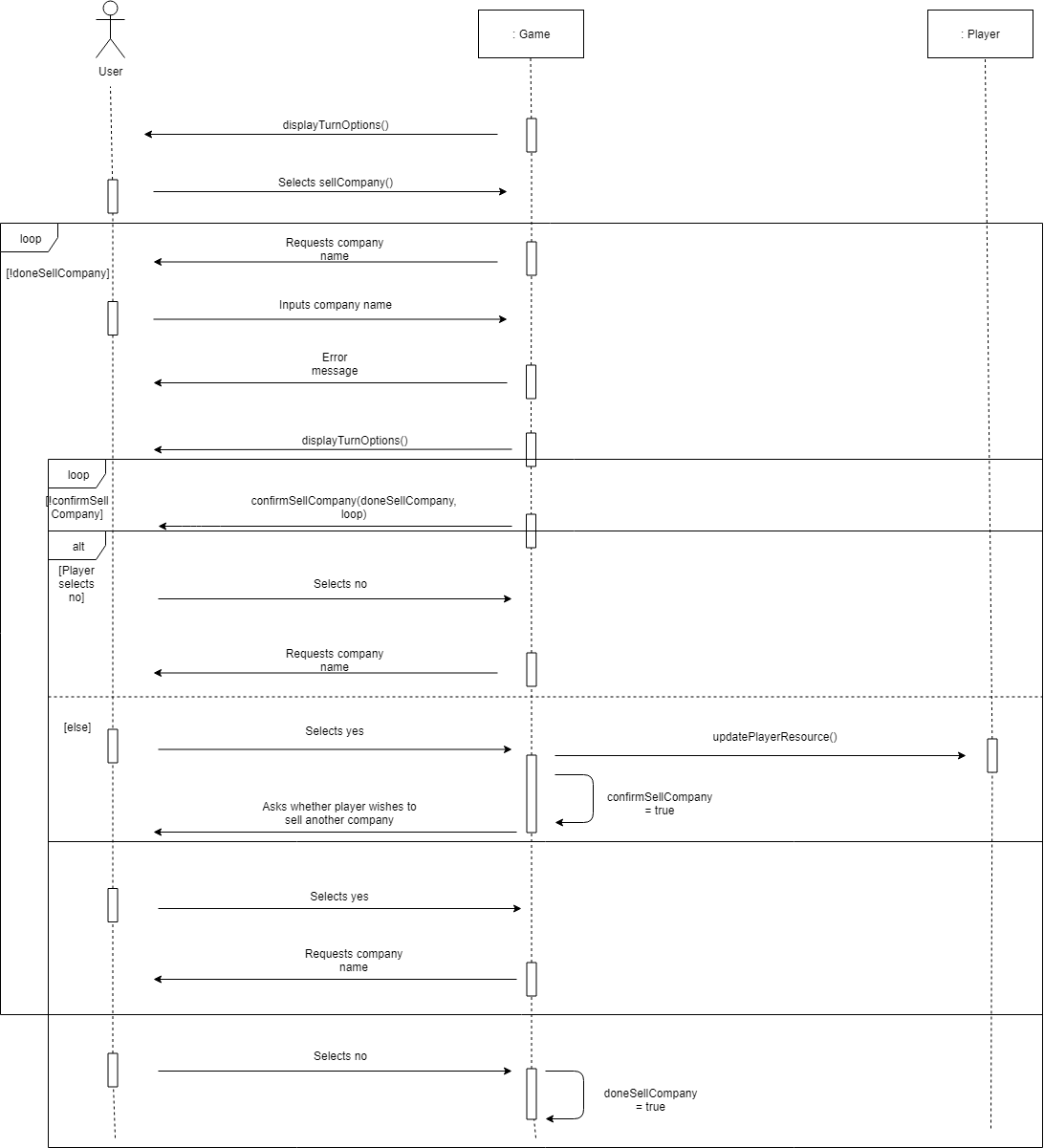
**5. Movement and Display Square Details.** If a player selects rollDice() from the displayTurnOptions() menu, then two dice are rolled to produce a number between 1 and 12. updatePlayerPosition() is called to move the player the requisite number of spaces and updateCurrentSquare() is updated to contain the square on which that player has landed. currentSquare.sendSquareDetails() is then called to produce the square’s details on console for the user and to begin any associated menu logic, such as buying the company on the square/paying rent (see below).



**6. Pay Rent and Buy Company.** Once a player lands on a square, if it is unowned then the buyCompany() method will be called to give the user the opportunity to buy it. If they opt not to buy it, then endTurn() is called; otherwise, setSquareOwnership() is called to change the square owner to the player and updateResource() is called to update that player’s resources.

If the square is already owned, then updateResource() is called to deduct the subscription owed to the square owner, depending on the number of offices/campuses on the square.

If the player’s resources drop below zero as a result, endGame() is called.

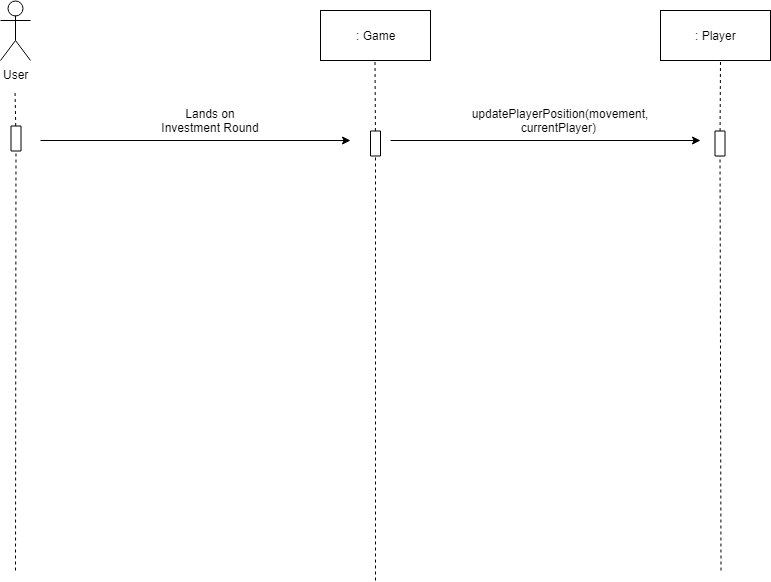


**7. Sell Company.** If a user selects sellCompany() from the displayTurnOptions() menu, then a series of system outputs are looped until the player indicates they are done selling companies.

The user is asked to input the company name, which is checked against the companies owned by the player. If the input does not match the name of any owned companies, then an error message is displayed and the loop continues.

If the user input does match an owned company, the user is asked to confirm sale of this company. If they select no, then they are returned to the top of the outer loop. If the player selects yes, then updatePlayerResource() is called to update the player’s resources accordingly.

confirmSellCompany is set to true, closing the inner loop requesting confirmation of the sale. The system outputs a message asking if the player would like to sell another company. If they select yes, then the outer loop continues. If they select no, then doneSellCompany is set to true and the loop closes, returning the player to the displayTurnOptions() menu.



**8. Investment Round.** updatePlayerPosition() is called during each movement. If the player passes the Investment Round during this repositioning, then they are given more resources.