Documentation

For Battleship

Team Defenders

-Md Saidur Rahman

-Sahat al Ferdous Fahim

-Md Limon Apu

-Moshiour Rahman Prince

Table of contents

1. Employed tools & technology

This game is programmed with c language. We have used two types of IDE,

1.1 Visual studio code.

1.2 Code blocks.

Software engineering part also required UML diagrams. We used MS Visio professional for that part.

2. Overall architecture

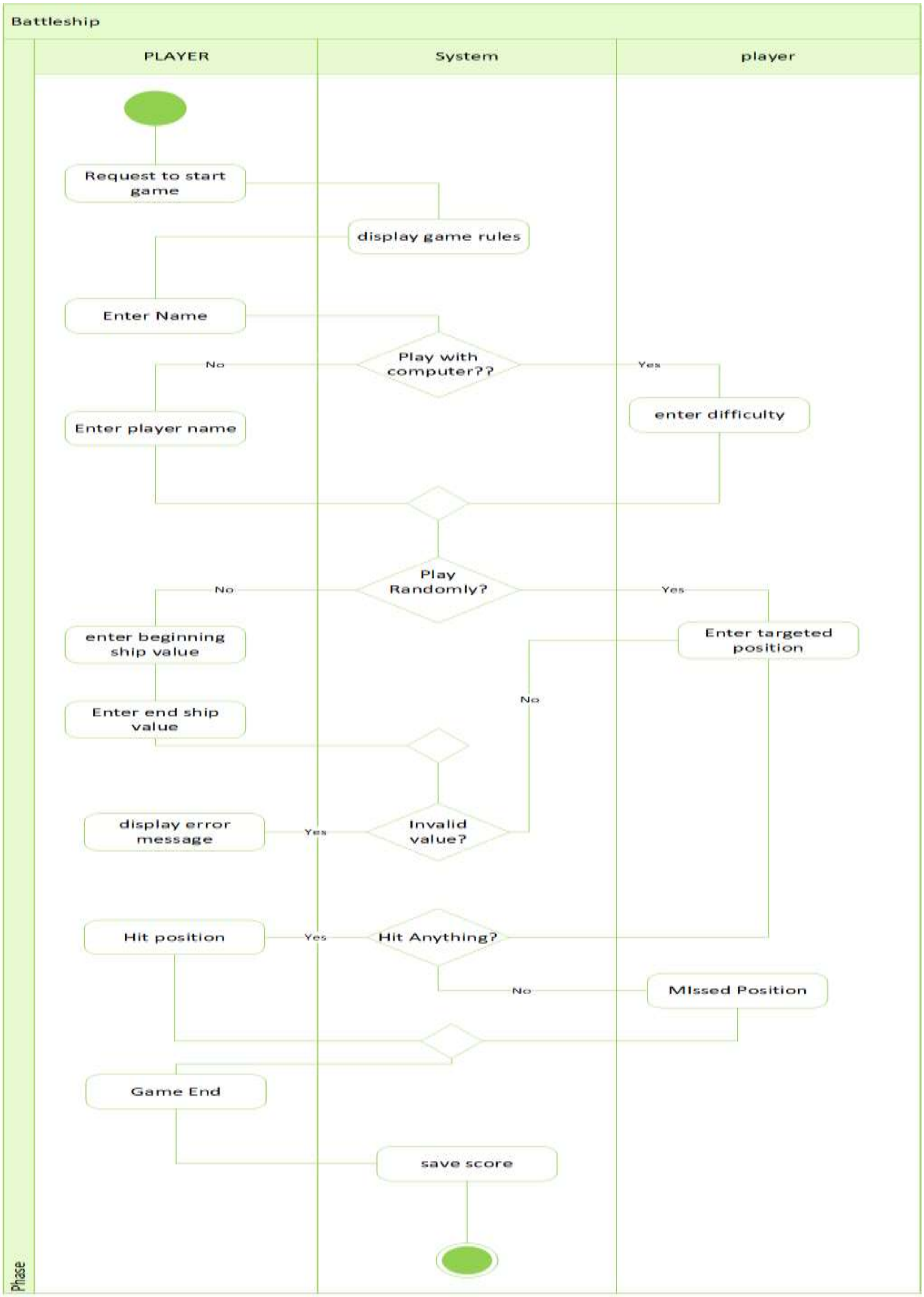
We have used 12 functions (excluding main) in order to implement this game. We will discuss important functions in important implementation details. In this section an overview of the game is discussed with a sequence and activity diagram respectively.

2.1 Activity Diagram: At the beginning the user has to start a new game. Then we display game rules. User chose a game name. Now there are two scenarios either with a computer or another human player.

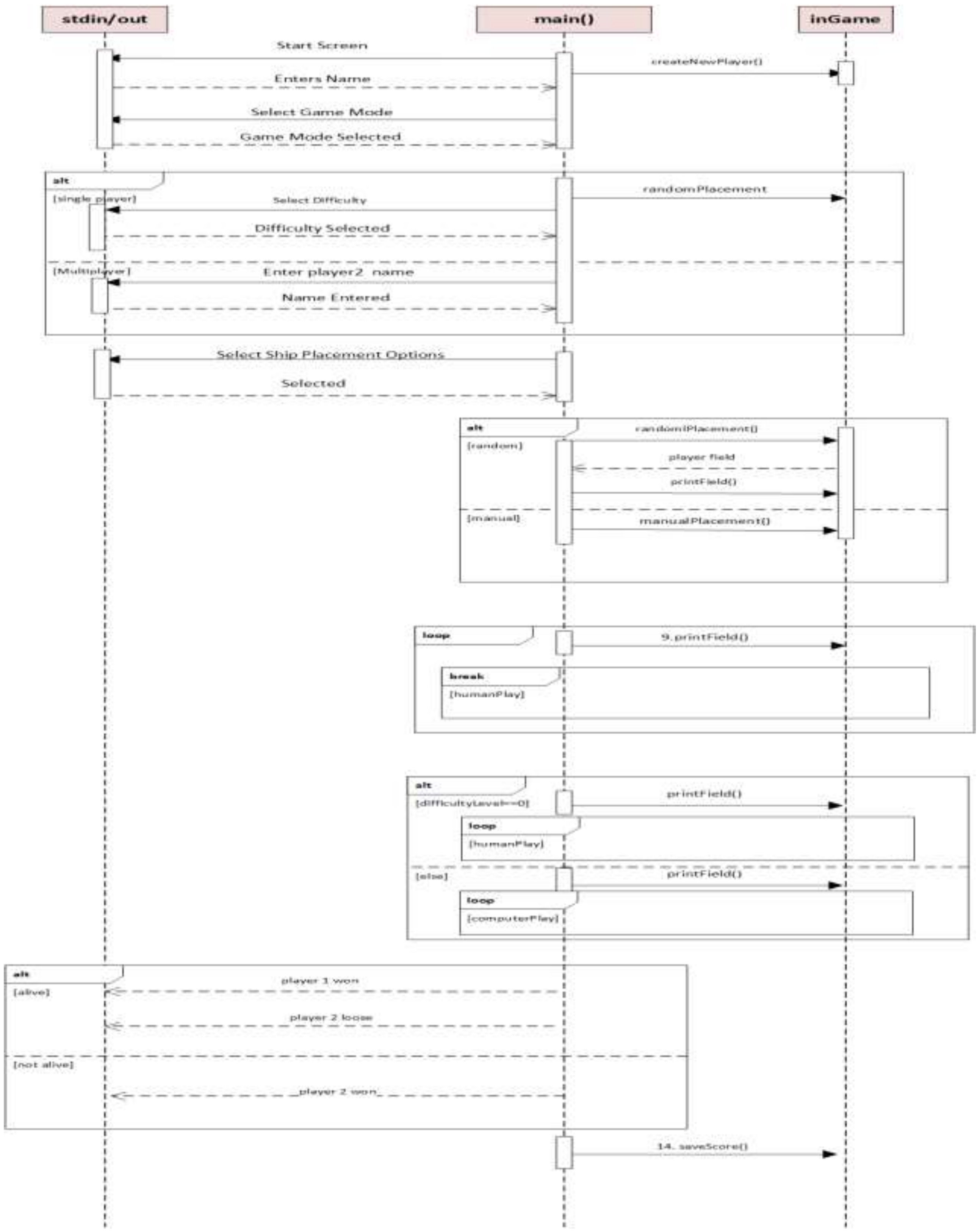
If user selects computer mode he is asked to select a difficulty level. Game is played against computer. User has the option to either place the ships randomly or manually. User will be asked to attack first. If he hits he will get to shoot until he misses a shot. Based on the difficulty level AI will attack and destroy user ships.

If user wants to play against a human player. He is directly asked to place the ships either randomly or manually. After placing both player's ships player 1 gets to hit first. Game continues until one of the player's all the ships are destroyed.

In both scenarios, there will be a guess count. This guess count will be saved as a score in a text file.


Battleship
PLAYER
System
player
Phase
Request to start game
display game rules
Enter Name
Play with computer??
No
Yes
Enter player name
enter difficulty
Play Randomly?
No
Yes
enter beginning ship value
Enter targeted position
Enter end ship value
No
display error message
Yes
Invalid value?
Hit position
Yes
Hit Anything?
No
MIssed Position
Game End
save score

2.2 Sequence Diagram



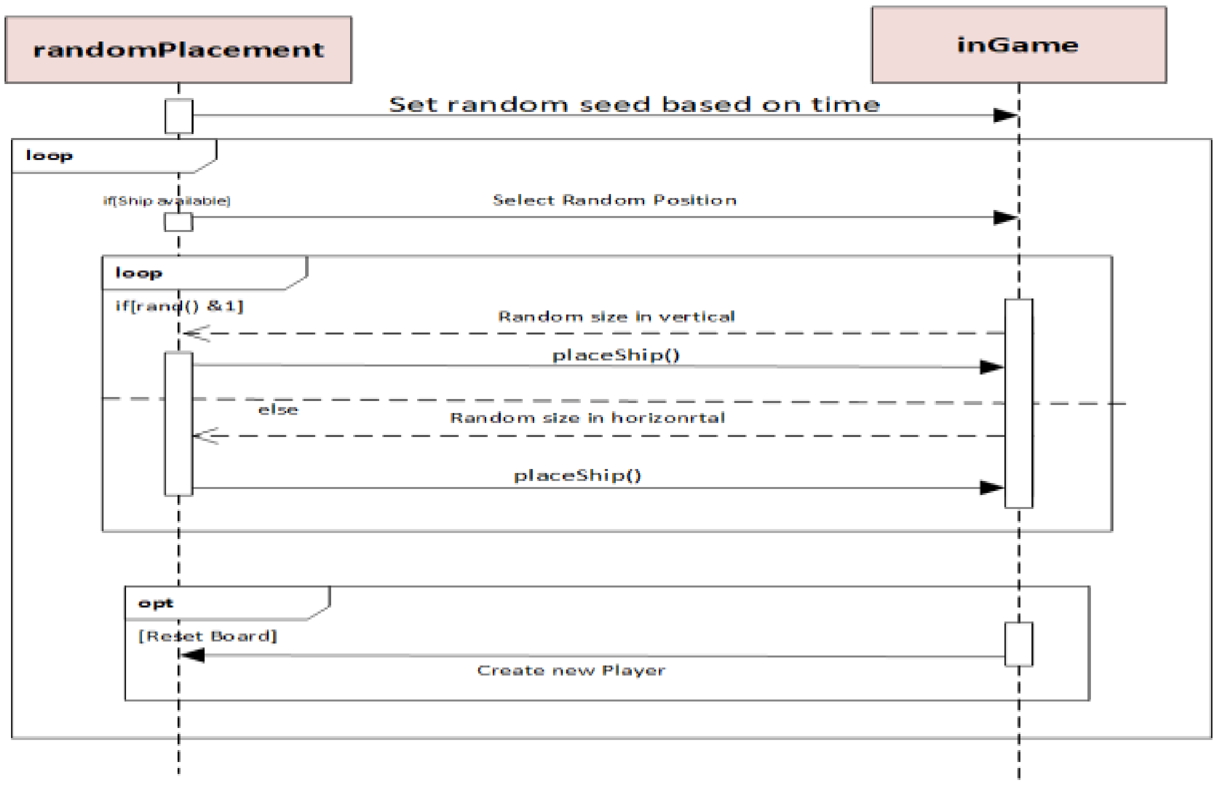
3. Major Design decisions

- We created a struct at the beginning to store information about player. By using this struct accessing player's field or other information was very easier. Combination of pointers and struct was very helpful.
- Using of functions smartly in order to keep the code DRY was also an important decision to make. For example, in this game field is printed over and over. In order to keep code lines minimum, we identified the parts which will need to be used repeatedly, we generated an algorithm and implemented them in a function.
- We implemented computer AI based on computer generated random numbers. We also used an important algorithm, If the computer doesn't hit first time after fulfilling a certain condition it hits multiple time.

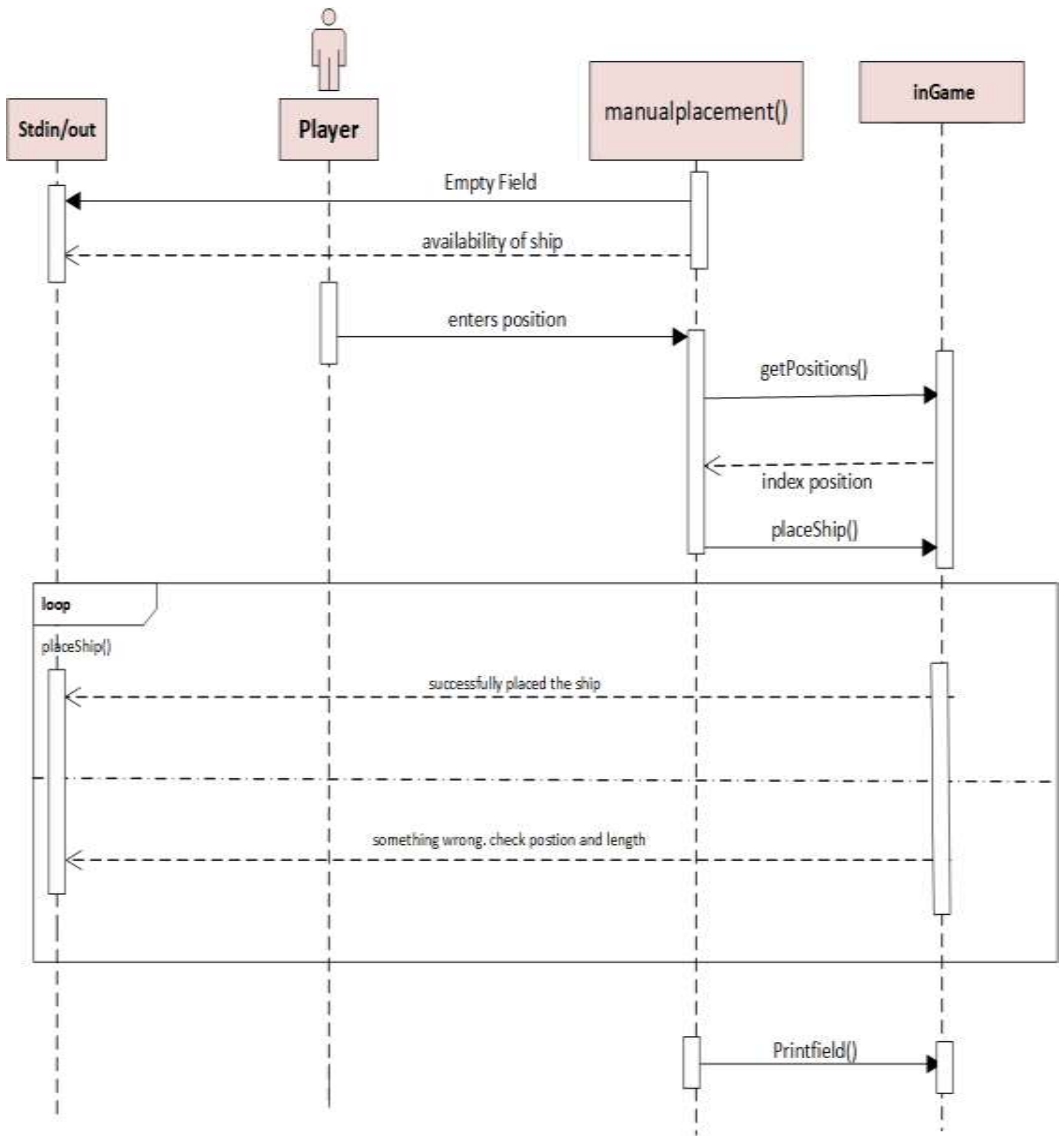
4. Important Implementation Details

This part will provide detail information about our important functions in the game with sequence diagram.

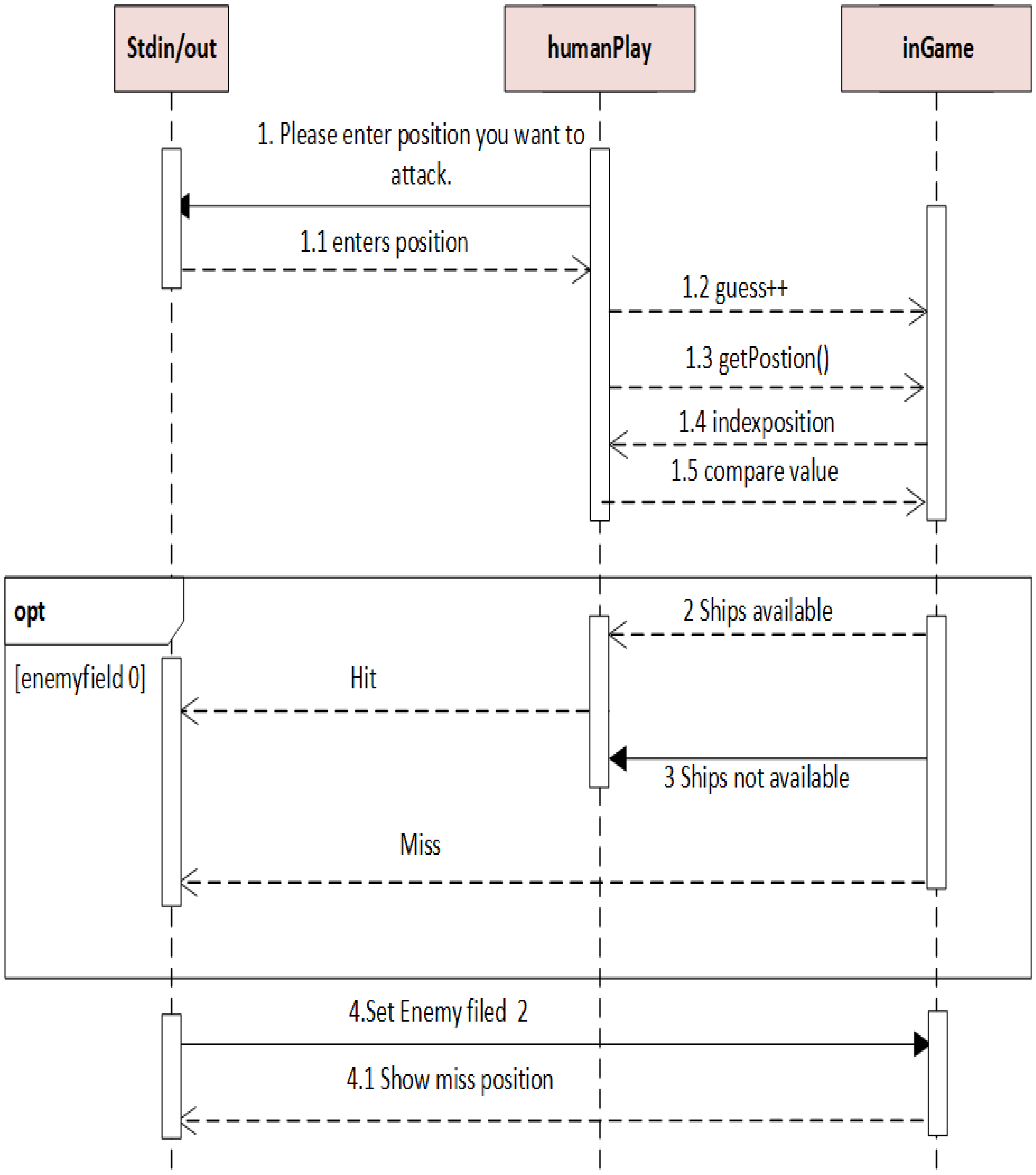
4.1 Random placement: This function is called when the player wants to place ships randomly. It places the ships both in horizontal and vertical direction. Function ends with another function create new player.



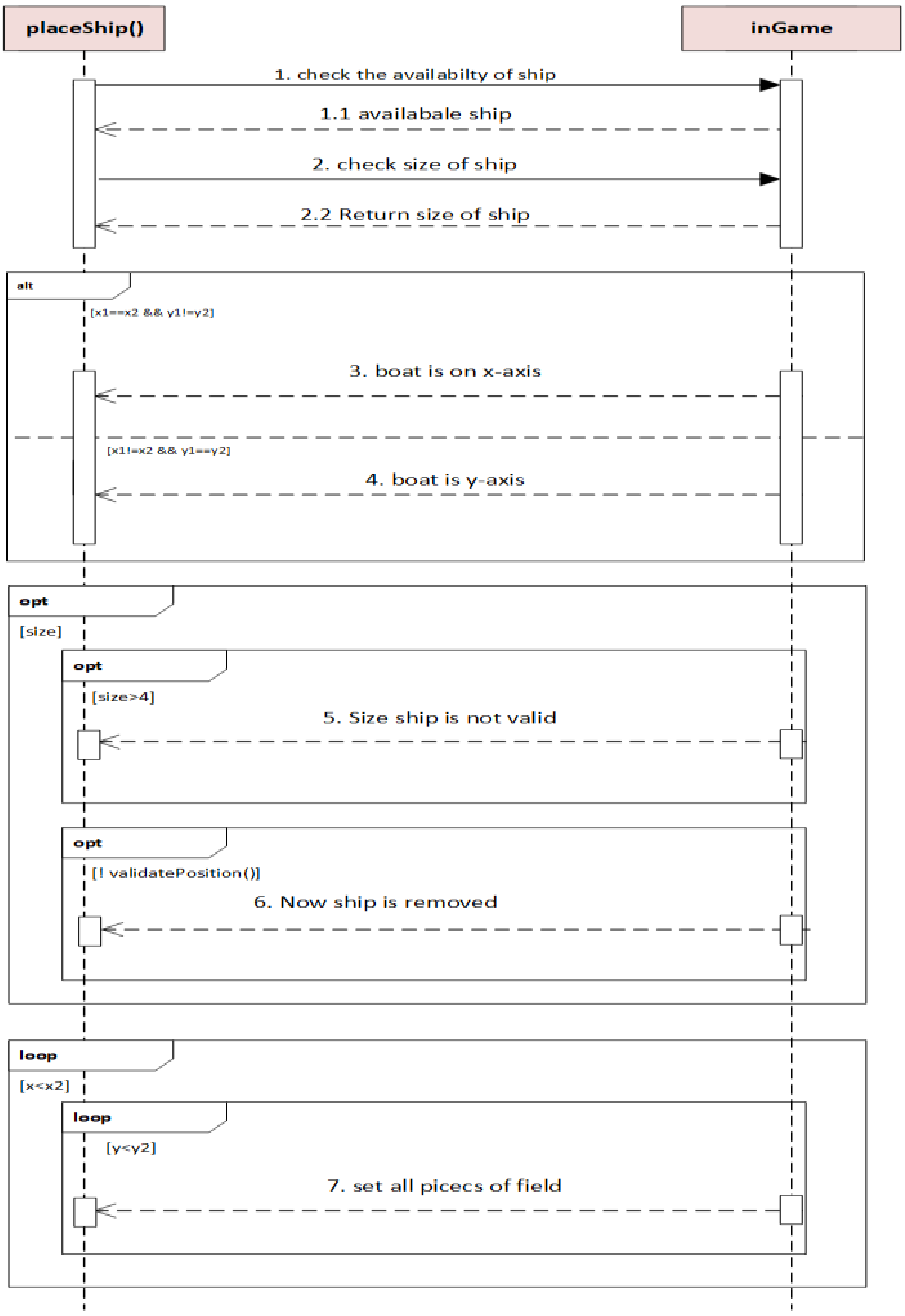
4.2 Manual Placement: This function is called when the player wants to place ships manually. It starts with empty field and total number of available ships. It asks the user to insert start and end position of the ship and uses another function called *getPosition* to convert the user input to player field indices. Then place ship function verify inputs using *validatePosition* function. If the position is valid the ship placement is successful. After each successful placement the function calls *printField* function to show the placed ship in the terminal. The function continues in a loop until all available ships are successfully placed. It ends with printing a final grid for the user.



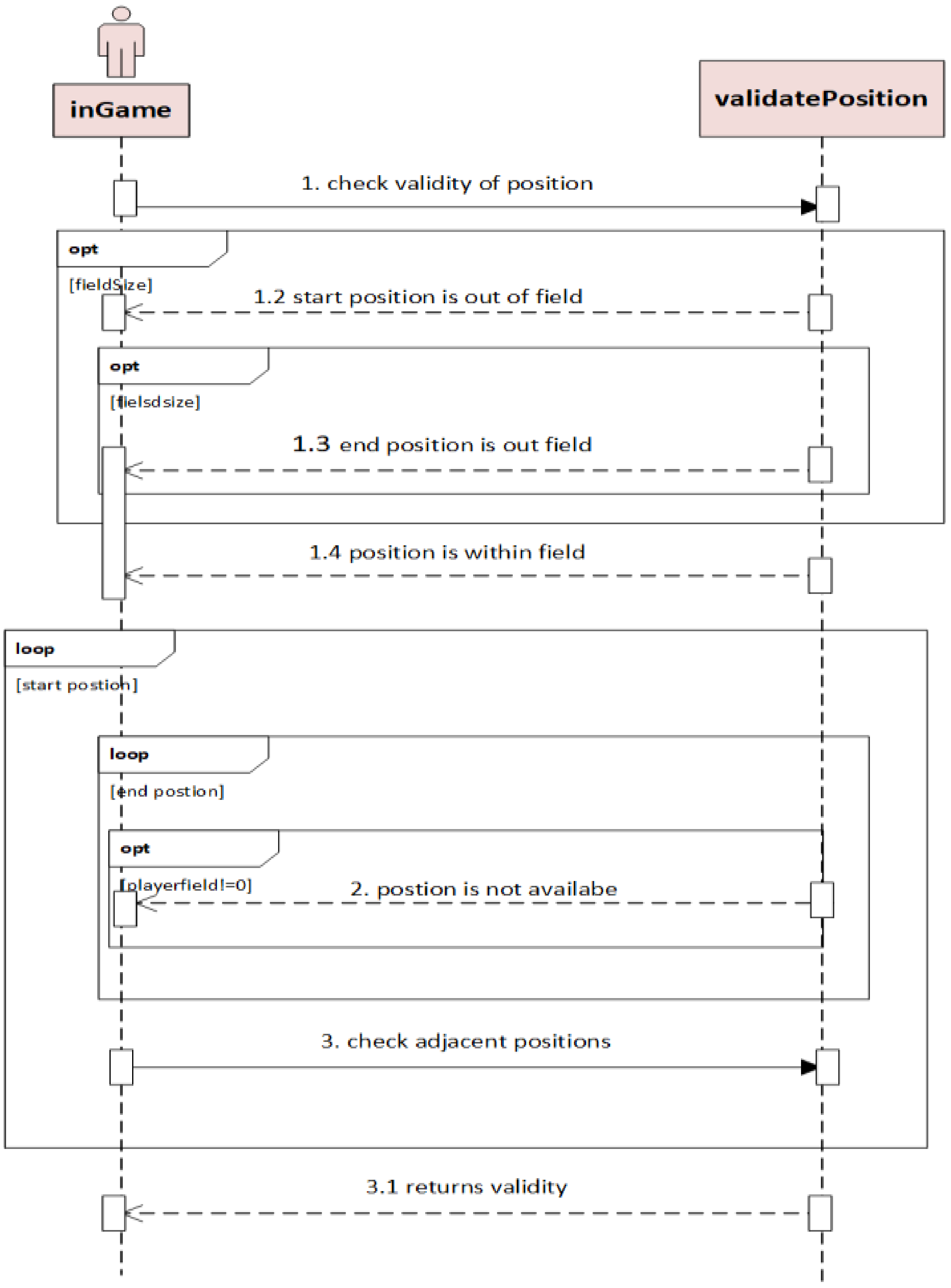
4.3 Human Play: This function plays the main role in user's gameplay. It counts and save how many chances did the user needed to finish the game to save it as a score. This function is mostly used implement if it's a hit or a miss. If user hits multiple times he gets multiple chances to shoot.



4.4 Place ship: It is function important to add ships to the grid. It checks the length and availability of the ship adds it to the field. It also shows how many ships are left to be placed.

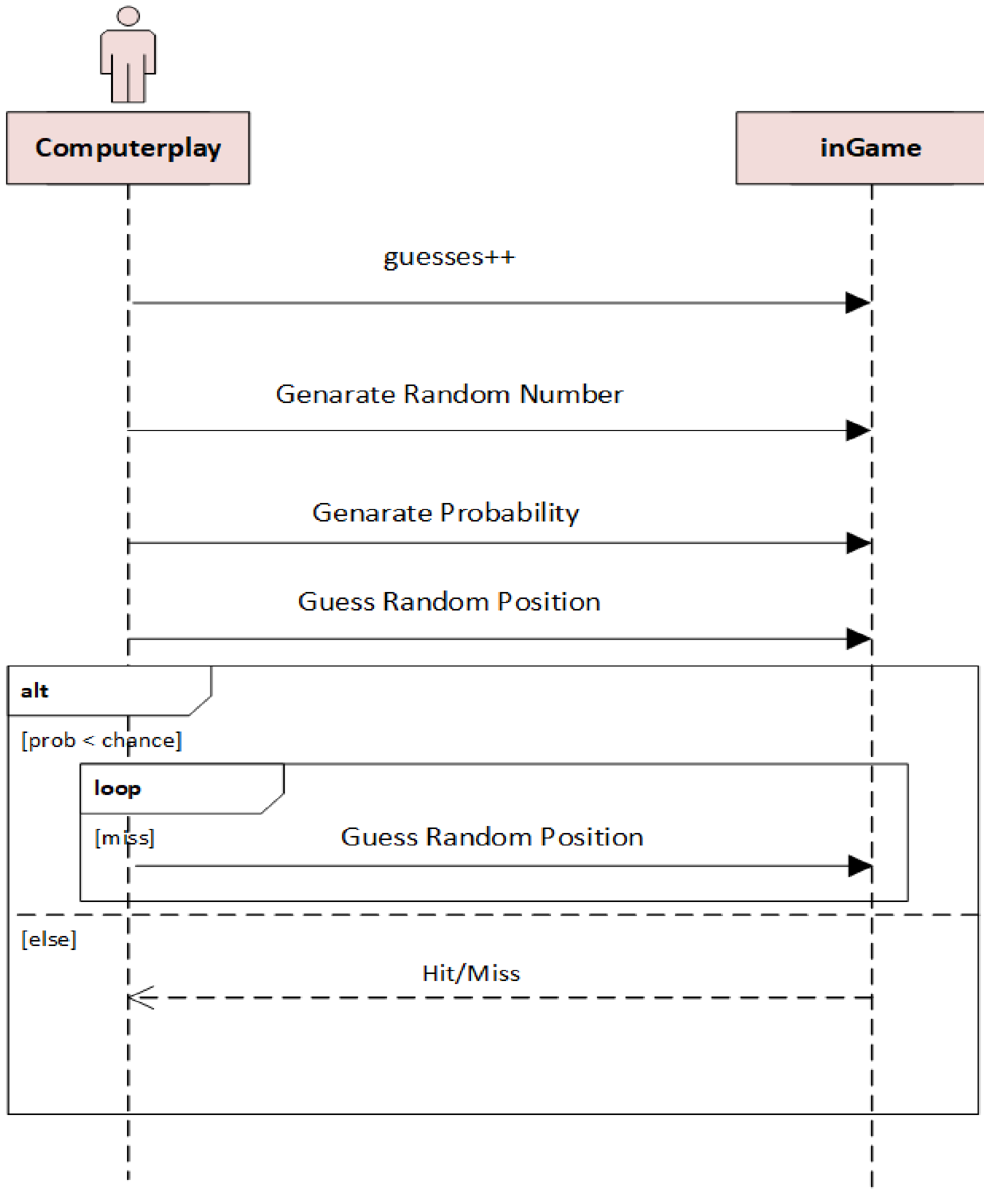

placeShip()
inGame
1. check the availabilty of ship
1.1 availabale ship
2. check size of ship
2.2 Return size of ship
alt
[x1==x2 && y1!=y2]
3. boat is on x-axis
[x1!=x2 && y1==y2]
4. boat is y-axis
opt
[size]
opt
[size>4]
5. Size ship is not valid
opt
[! validatePosition()]
6. Now ship is removed
loop
[x<x2]
loop
[y<y2]
7. set all picecs of field

4.5 Validate Position: This function is used basically to implement our requirements. It validates the position of the ship. For example, it verifies if the ship we placing is either vertical or horizontal. Ship overlapping is avoided by this function.



inGame
validatePosition
1. check validity of position
opt
[fieldSize]
1.2 start position is out of field
opt
[fielsdsize]
1.3 end position is out field
1.4 position is within field
loop
[start postion]
loop
[end postion]
opt
[playerfield!=0]
2. postion is not availabe
3. check adjacent positions
3.1 returns validity

4.6 ComputerPlay: First the computer generates a floating number called chance by dividing the difficulty with 6. Then the computer generates another random number ‘prob’ which is a random floating number from 0 to 1. After that computer generates random position to shoot, the AI checks if the computer previously targeted there, If its true then AI generates a new random number again. Then the AI checks if the random floating number is smaller than ‘chance. If this condition is true then the computer targets until it hit something. If this condition is false then the AI checks if the targeted position is a hit or miss.



5. Organization

- **Week 1:** We started by analyzing our task and created an algorithm. Then we marked out the parts that we might need to use multiple times and gave them name as functions.
- **Week 2:** After that we had to research about finding solution to those functions that we do not know how to solve. In this period, we had to learn deeper about using struct, pointers, loops, file creation, file editing without opening the file.
- **Week 3 & 4:** In this two weeks we focused on writing our code.
- **Week 5:** Writing the documentation of our project started here. In this week we also worked in our code to find solution for error handling and reconstructing extra lines of code in a simpler manner.

Which member of the group was responsible for which part of code is given in the table below

Name	**Implemented Funtions**	**Overall contribution in %**
Md Saidur Rahman	*main, createNewPlayer, printField, alive.*	25%
Sahat al Ferdous Fahim	*randomPlacement, randomRange, _sum, computerPlay, humanPlay.*	25%
Md Limon Apu	*humanPlay, manualPlacement, getPosition, computerPlay.*	25%
Moshiour Rahman Prince	*validatePosition, placeShip, saveScore, Comments.*	25%

Which member of the group was responsible for which part of documentation is given in the table below

Name	**Contributions**	**Overall Contribution in %**
Md Saidur Rahman	*main*-Sequence Diagram, m*anualPlacement*-Sequence Diagram.	25%
Sahat al Ferdous Fahim	*randomPlacement*-Sequence Diagram, *placeShip*-Sequence Diagram.	25%
Md Limon Apu	*main*-Activity Diagram, *computarPlay*-Sequence Diagram.	25%
Moshiour Rahman Prince	*validatePosition*- Sequence Diagram, *placeShip*-Sequence Diagram.	25%