**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

Role of Artificial intelligence plays major role many fields especially in the areas where the human intervention or human error is to be avoided. This need not even left the gaming field also ,the reason is people are tired of playing with traditional computer opponent so they wanted to play with the human opponent so there came the need for multiplayer online and offline games ,But later the online multiplayer games where over flown with online players. This too didn't last for long because of the players busy schedule players who want to play in their free time didn't find the other players online ,so there came a need for the AI to act in the place of human player. This project is a multi player snake game in which two player compete each other on a single board.

This AI substitute of the human player should allow the opponent players to win as well as to loss . So the AI should be build in such a way that it plays like a actual human player. This project has developed a AI to capture the food and to escape from the opponent's attacking moves and patterns he follows to defeat his opponent.

The bigger challenge in this project is the data collection from the user and facilitating the players to player with their friend ,So thinking of the better way to connect the people the first thing that comes to our mind is the web technologies which has played and being playing a major role many field like banking ,entertainment ,medicine and education. So a better platform to implement this idea is the web technologies

Now the disclosed information is that the project is entire depend on web technology ,but how the a web technology is going to serve a need of the gaming project because the gaming need a lots of data to be transferred over the network for smooth and clean gaming experience with the friends and strangers.

This need is satisfied by the many server push mechanisms ,one such mechanism formulised is SSE(Server Sent Events). The next challenge is the data storage and collection .The better way to store the data in a web environment is the database ,this project uses the MySQL database to store the data and also for retrieval ,manipulation and updation of data.

**1.2 MULTIPLAYER SNAKE GAME**

This Snake game is mainly focused on the multiplayer mode rather than the single player mode. The major challenge of the multiplayer mode over single player mode is the end point of the game .In single player mode the end point of the game is none ,it depend on the player's ability and only the score matters but in multiplayer mode the end point is entirely different ,any one of the snake should stand or both should fall so the end point is led by attacking each other ,even if they don't attack the game board cannot accommodate their body after they eat several food. In this multiplayer concept the player can attack each other ,if one player attack the other at particular body part then the parts after that will be discarded ,in this way their body part will be reduced and the one with body part less than the desired level will be declared as lost and the other as winner. Here a doubt may arise, then how will they develop their body part .This operation is done by consuming the food displayed in the board. This food is also distributed in synchronised manner that is both the players will get same food at same co-ordinates and there is no different weightage for different food, every food will have same weightage .

**1.3 CHARACTERISTICS OF THE MULTIPLAYER SNAKEGAME**

Received moves as result of the several games played by the player among their friends more and more amount data were got accumulated in the database which could be properly maintained and manipulated to give excellent user experience .This data is collected while the player gives any moves to their snake in any given situation .Thus the collected data is used by the computer during the game play with the other human player ,for example consider both the player and the computer are approaching for a food now the human player will have clear idea of what he/she should do but a computer will not have such an idea or do not have any prediction of the opponent’s next move ,so to make the computer comfortable with the opponent human player these data gives wide view for the computer. Some more characteristics that are added are to this game is that the computer snake reveals its current state that is it change its snake colour when it can predict the opponent’s next move and change to another colour when it cannot predict the opponent’s next move. The reason to have this feature is to increase the curiosity of the opponent player so that his interest in the game increases and he/she give their valuable move data to computer for its future use.

This game follows the steps below:

1- Collect the data from the user by a multiplayer game session.

2- Do some prediction job when a situation is met .

3- Uses the predicted data to escape from the opponent.

**1.4 SERVER PUSH MECHANISM FOR MULTIPLAYER GAME**

The first model proposed is entirely a server oriented model in which the entire game data is maintained in the server side and continuously sent as a event stream to the client, the major drop back of this model is the efficiency in term of both time and space occupied by the data transferred over the network. The reason for this failure is due to the enormous amount of object created by the both client and server. This failure is overcome by the second model which completely localized the to the individual client machines by the client agent like browser. This model has improved the efficiency of the client as well as the server which should serve the several similar clients. So here come the need for the synchronization because the game is left to the individual client and only some data are sent to the server regarding the multiplayer session.

**1.5 PREDICTION FOR BOT**

In this project the prediction is done to guess the human opponent’s next move and make a relevant action to the computer snake. This prediction is basically done by calculating the likelihood of any given classes. This game involve totally four classes they are left ,right ,top and down .These four classes are classified based on eight parameters they are head position of the snakes and distance between food and snake and length and current direction of the snake.

**1.5.1 Database for Prediction Model**

**Moves database**

This is the database which contains various moves made by the player in their previous games, the attributes of the subject relation are headx1,heady1,l1,d1,id1,id2,d2,l2,dixs,diyx,dix1f,diy1f and game board id.

**CHAPTER 2**

**LITERATURE SURVEY**

|  |  |
| --- | --- |
| **2.1** | **SLITHER.IO** |

***Slither.io*** is a massively multiplayer video game available for iOS, Android, and web browsers, developed by Steve Hawse. Players control an avatar resembling a worm, which consumes multicolored pellets, both from other players and ones that naturally spawn on the map in the game, to grow in size. The objective of the game is to grow the longest worm in the server. *Slither.io* is similar in concept to the popular 2015 web game *Agar.io* and is reminiscent of the classic arcade game *Snake*.

The game grew in popularity following its promotion among several prominent YouTube users such as Pew Die Pie, and topped the App Store shortly after its release. *Slither.io*'s browser version was ranked by Alexa as one of the 1,000 most visited sites by July 2016, while the iOS version ranked first in the most downloaded apps on the App Store. A mobile version of the game for Android was released on March 27, 2016. The reception of the game was positive, with reviewers praising its appearance and customization but criticizing it for its low replay value and the high price users must pay to remove advertisements.

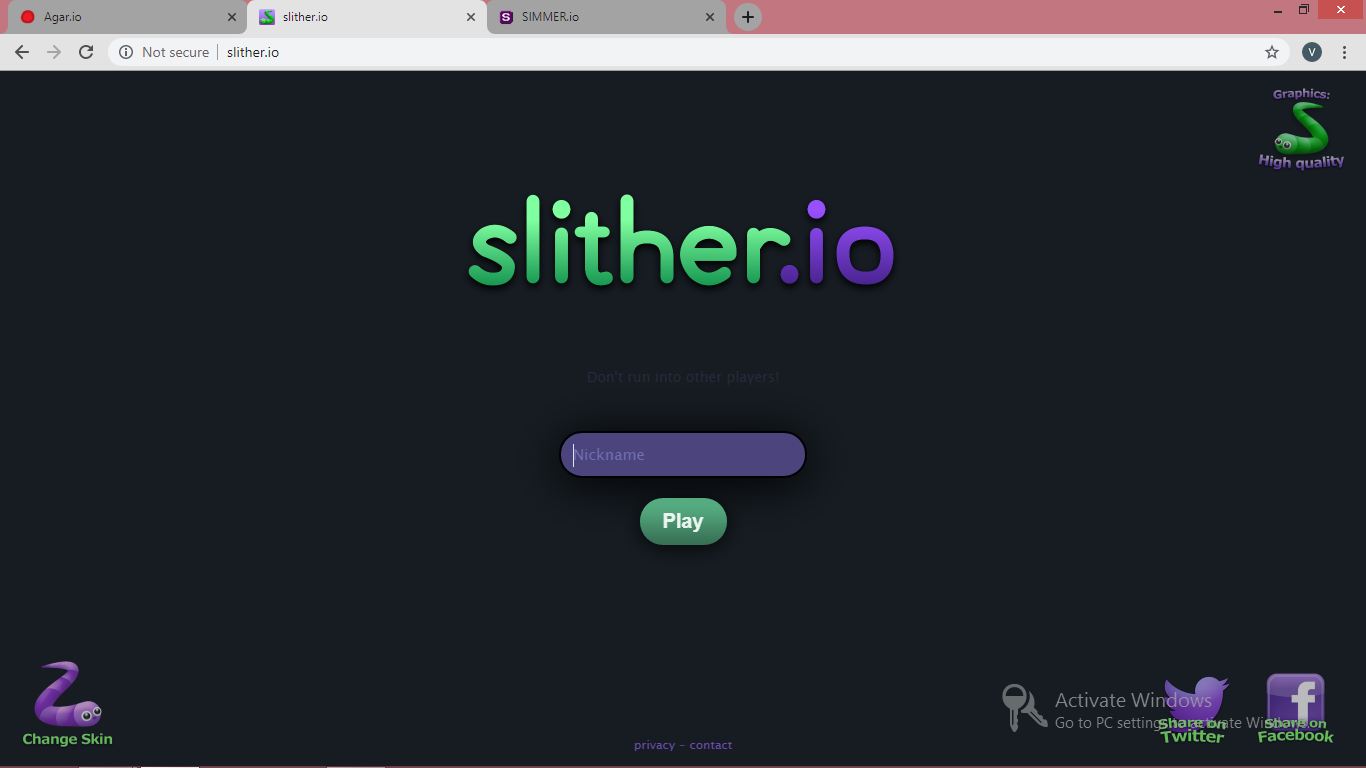
**2.1.1 Development**

According to game creator Steven Hawse, he was inspired to create the game while he was experiencing financial problems. As a result of these financial issues, he had to move from Minneapolis to Michigan, where he realized the popularity of *Agar.io*. He had long wanted to create an online multiplayer game, but the only option for the development at the time was in Adobe Flash, and not wanting to use this method, he gave up the idea for a while. Hawse finally created the game when he realized that Web Socket, a low-latency protocol supported by most major browsers, was sufficient and stable enough to run an HTML game similar to other games such as *Agar.io* The most difficult part of the development was in making each server stable enough to handle 600 players at a time. Hawse struggled to find space on servers with enough space in regions where there was more demand and tried to avoid cloud services like Amazon Web Services, owing to the high cost that these services would incur based on the amount of bandwidth used .

After six months of development, *Slither.io* was released for browsers and iOS on March 25, 2016, with servers supporting up to 500 players Two days after the iOS/browser versions' release, an Android version was made available by Low-tech Studios. The only way Hawse could make revenue was to display advertising in the app after the player's worm died; this option could be removed for US$3.99. He chose not to sell virtual currency or power-ups so that those who paid would not have an advantage over players who did not. As there was no money to advertise for the game, the only way to advertise was the various *let's plays* by players on YouTube, including Pew Die Pie, who had more than 47 million subscribers at the time .

* + 1. **Advantages and Disadvantages**
* More than two player can compete on a single board which pretty much larger.
* Disadvantage is that it is easy to kill other player and get killed by other players.

**2.1.3 Screenshots**

****

**Figure 2.1 Slither.io screenshot 1**

****

**Figure 2.2 Slither.io screenshot 2**

|  |  |
| --- | --- |
| **2.2** | **AGAR.IO** |

***Agar.io***is a massively multiplayer online action game created by Brazilian developer Matheus Valadares. Players control one or more cells in a map representing a Petri dish. The goal is to gain as much mass as possible by eating agar and cells smaller than the player's cell while avoiding larger ones which can eat the player's cells. Each player starts with one cell, but players can split a cell into two once it reaches a sufficient mass, allowing them to control multiple cells. The name comes from the substance agar, used to culture bacteria.

The game was released to positive critical reception; critics particularly praised its simplicity, competition, and mechanics, while criticism targeted its repetitive game play. Largely due to word of mouth on social networks, it was a quick success, becoming one of the most popular web and mobile games in its first year A Steam version was announced on 3 May 2015 (though never released as of May 2019), while the mobile version of *Agar.io* for iOS and Android was released on 24 July 2015 by Miniclip. *Agar.io* has inspired similar web games called ".io games", including games with a similar objective but different characters, and games that incorporate elements of other genres like shooter games

## **2.2.1 Development**

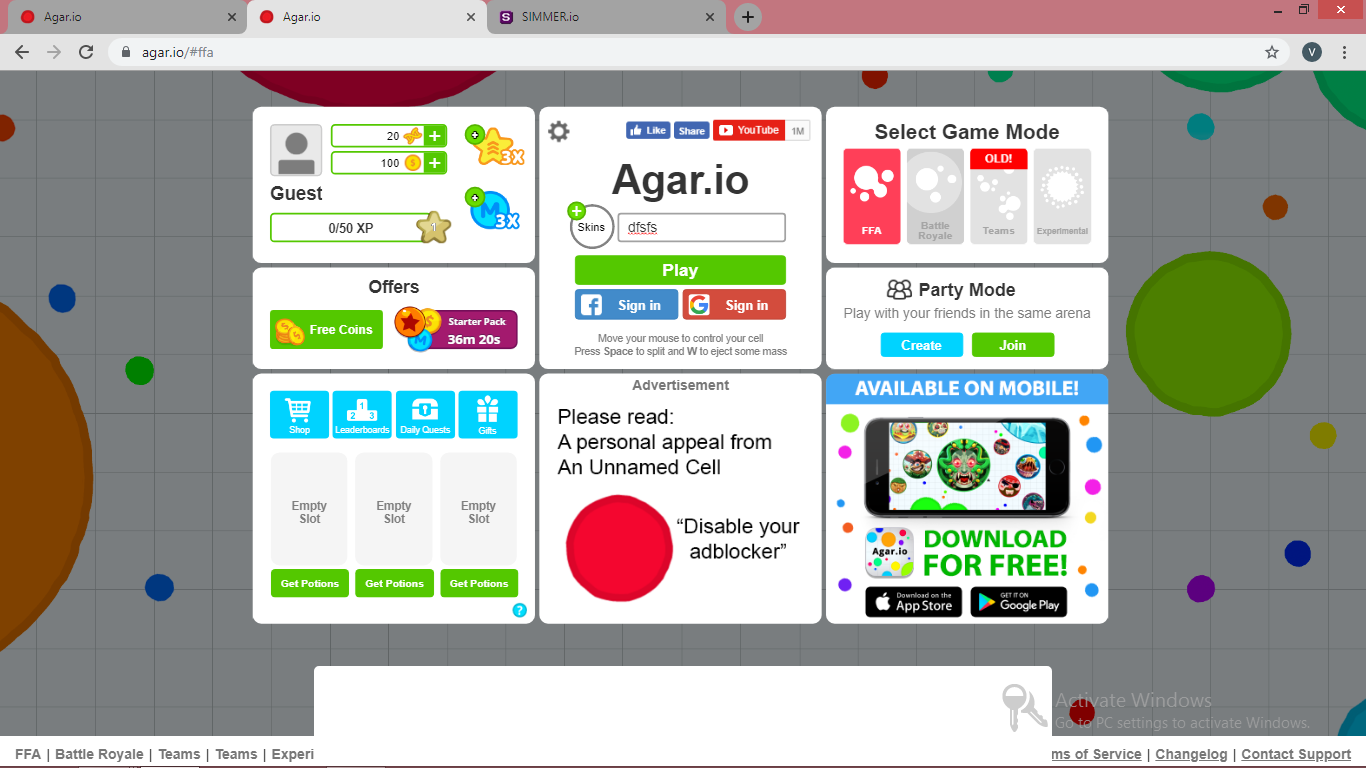
*Agar.io* was announced on 4chan on 28 April 2015 by Matheus Valadares, a 19-year-old Brazilian developer Written in JavaScript and C++, the game was developed in a few days The game originally did not have a name, and users had to connect to Valadares' IP address in order to play. The name *Agar.io* was suggested by an anonymous user on 4chan, as other domain names such as cell.io were already taken. Valadares continued updating and adding new features to the game, such as an experience system and an "experimental" game mode for testing experimental features. One week later, *Agar.io* entered Steam Greenlight with Valadares announcing a future free-to-play version of the game for download. He planned to include features in the Steam version not available in the browser version, including additional game modes, custom styling, and an account system. It was approved for listing on Steam due to community interest. However, the Greenlight program was shut down in 2017,[]](https://en.wikipedia.org/wiki/Agar.io#cite_note-18) and the announced game has yet to be released.

On 24 July 2015, Miniclip published a mobile version of *Agar.io* for iOS and Android. Sergio Varanda, head of mobile at Miniclip, explained that the main goal of the mobile version was to "recreate the gaming experience" on mobile, citing the challenges with recreating the game on touch screen controls

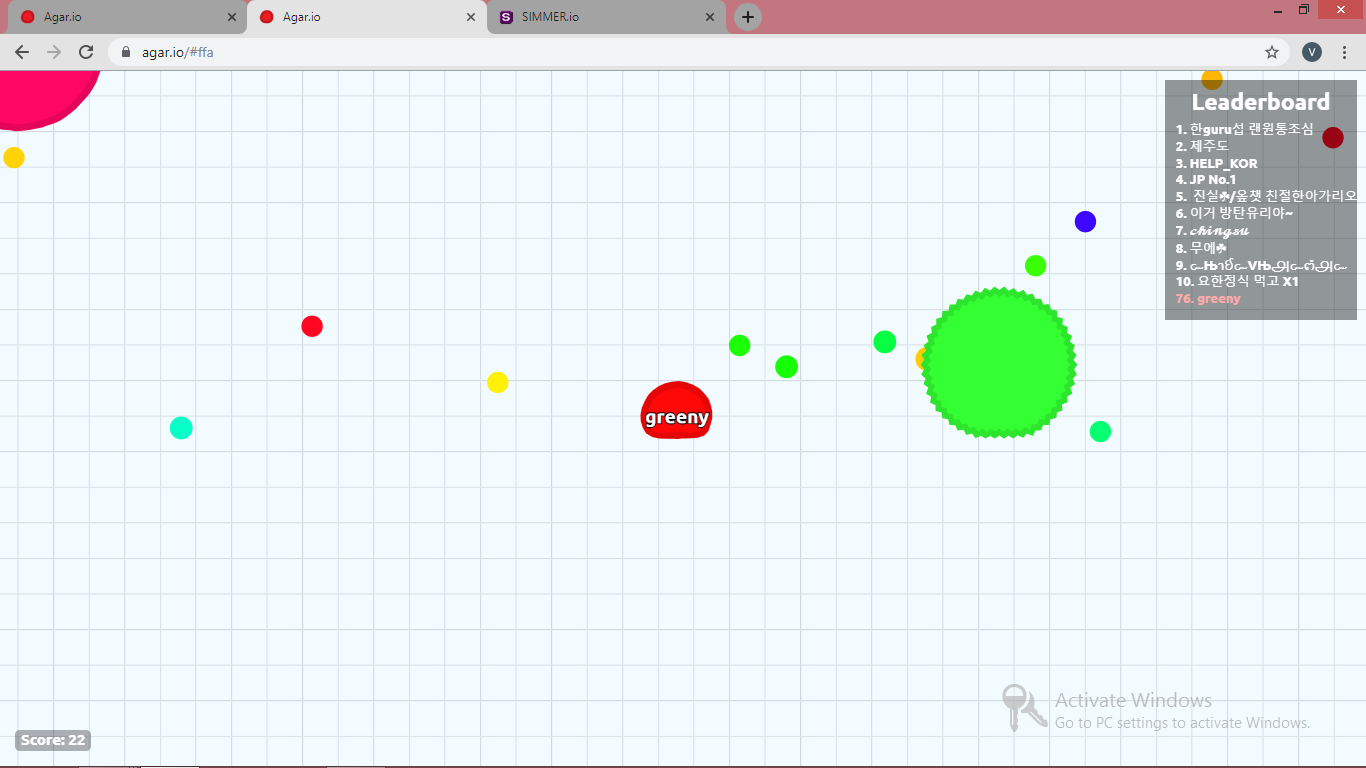
**2.2.2 Advantages and Disadvantages**

* It has more user accommodation capacity and improved input control.
* The basic functionality of the game is less.

**2.2.3 Screenshots**

****

**Figure 2.3 Agar.io screenshot 1**

****

**Figure 2.4 Agar.io screenshot 2**

**CHAPTER 3**

**PROPOSED WORK**

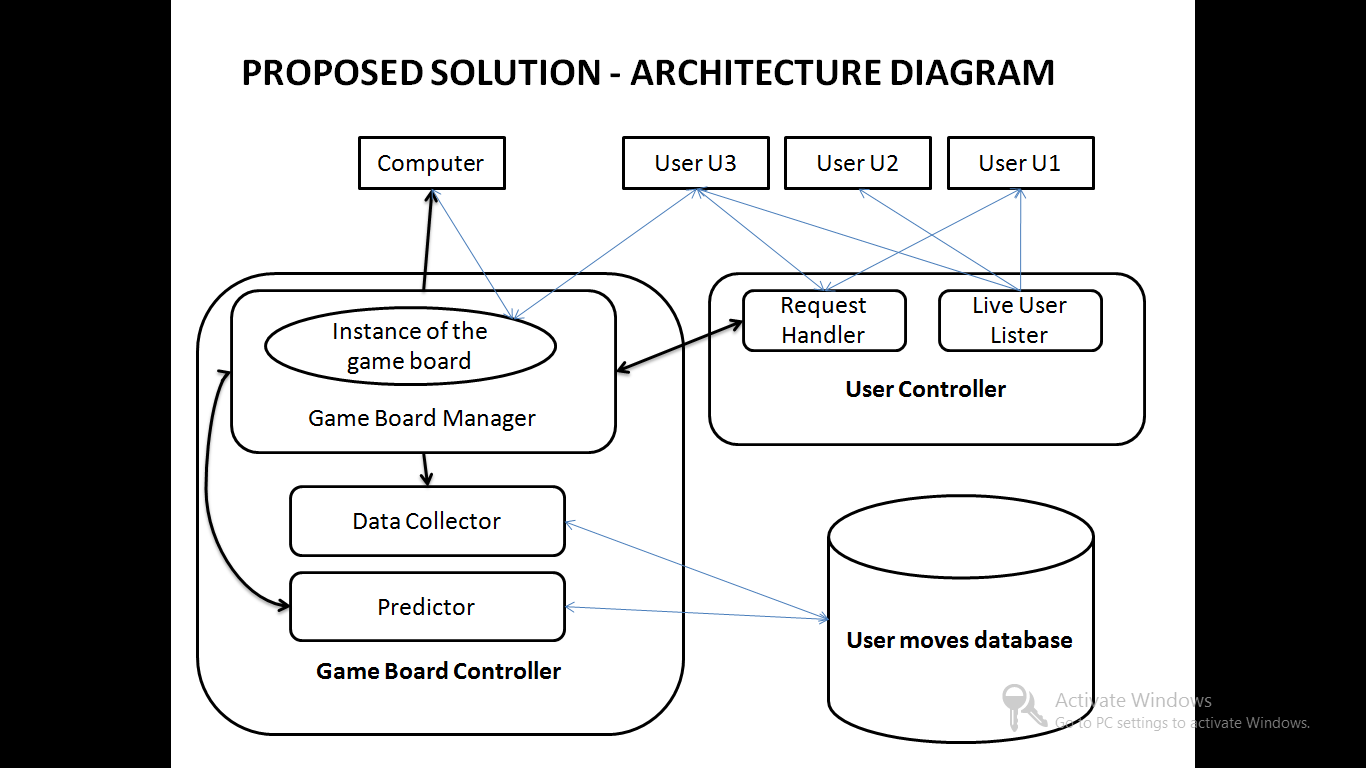
**3.1 RECOMMENDED METHOD**

Aim of this project is to develop a game to enable the player to meet online and search for person using his nick name and have a friendly game with him ,this purpose has been completely served by this game by a server push mechanism called the SSE and flag method to do the as multiplayer

**3.1.1 Advantages**

* The first advantage of this method is that it is light weight
* Usually any prediction system would require a basic dataset scrapped from somewhere but here there is no need for that since the entire data set has been generated by the customer .
* Anyone can play from anywhere by having the access to the network only ,this is possible through several networking job done in the system in deeper level.
* Thought there are no online players are available for a while there is no need for user to sit idle he/she can play a game with computer in the mean waiting while ,which will also improve knowledge base of the computer.
* The next big advantages of this model is knowledge capturing is done by the system automating when players are playing the game ,that is there is no separate predefined or trained data set is needed.
* Since the game is localized to the client machines the game is very much responsive that any other snake game available in .io game collection

**3.2 PROPOSED ARCHITECTURE DIAGRAM**

****

**Figure 3.1 Architecture Diagram**

**3.3 DESCRIPTION OF THE ARCHITECTURE DIAGRAM**

**3.3.1 Participants**

The participants of this game can either be a human or a computer since it has feature of single player against the computer and multiplayer mode which has player playing against a another player so the participant can be a human player or a computer.

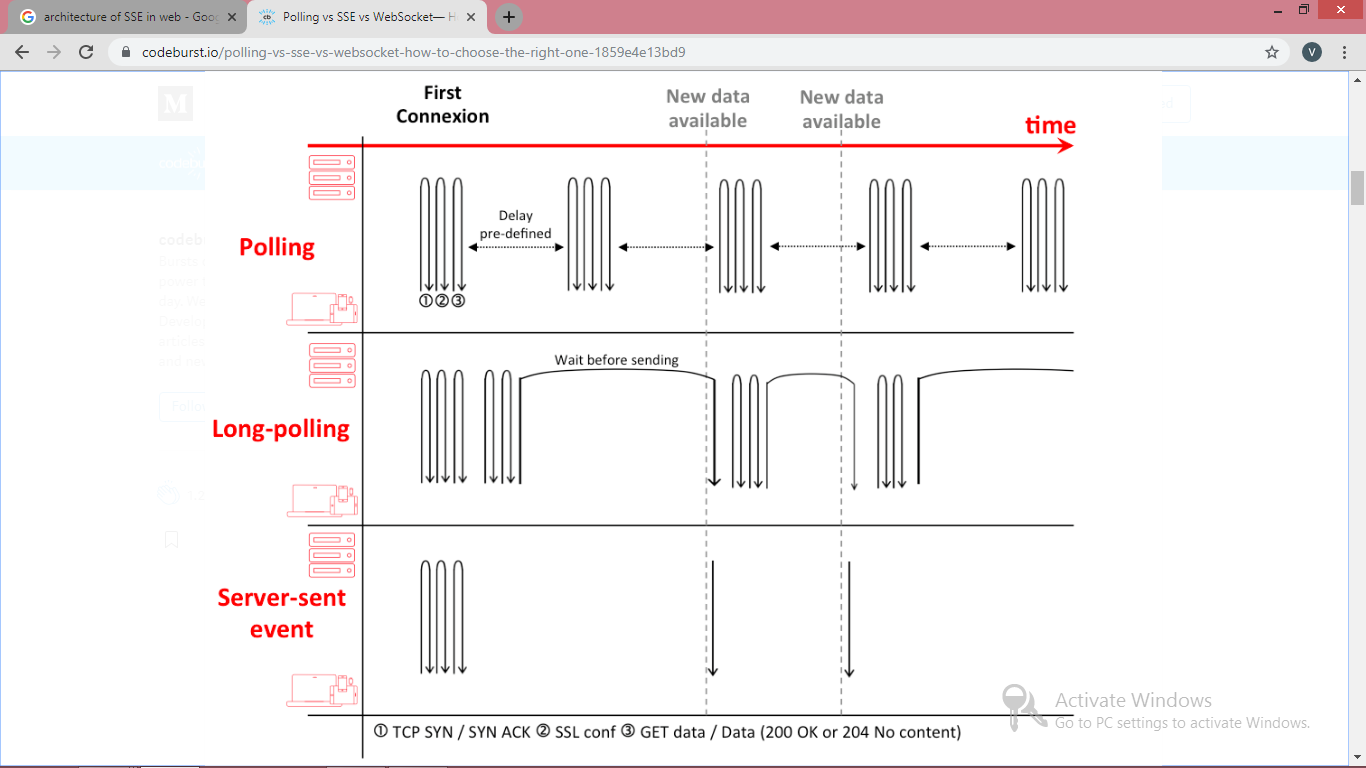
Every feature that involves the rela data flow from server to client has any touch of the HTTP server push job so let us first look into the mechanisms available for the HTTP server push job ,There are several mechanism developed for this purpose some of them are

* Server Sent Events
* Long/Short polling
* Web sockets
* COMET

Building a real-time web application is a bit challenging one, where we need to consider how we are going to send our data from the server to the client. Technologies that enable this “proactively” have been around for quite some time and are limited to two general approaches: *client pull*or *server push.*

A few ways to implement these:

* Long/short polling (client pull)
* WebSockets (server push)
* Server-Sent Events (server push)
* Client pull — client asking server for updates at certain regular intervals
* Server push — server is proactively pushing updates to the client (reverse of client pull)

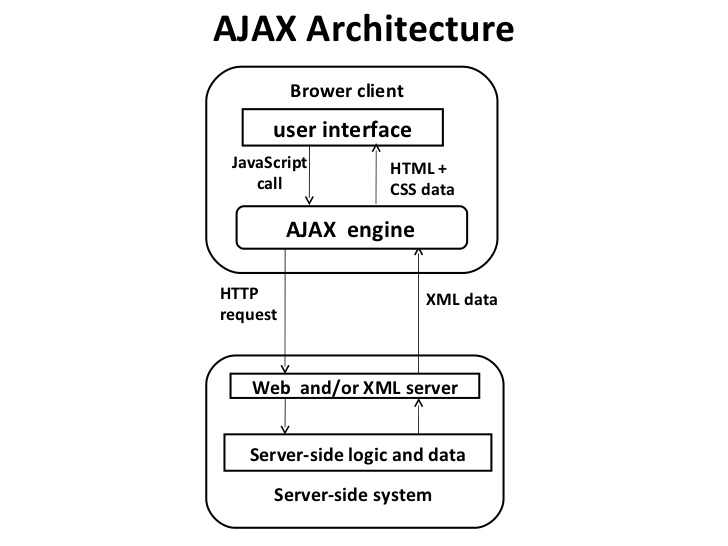


**Figure 3.2 Comparison of server push mechanisms**

For better user experience the request and response process should be carried out only on asynchronous manner ,that too in the background because if the browser handles these process then it will degrade the performance of the game .This degradation occur because in JS every process is a single threaded process so if we frequently send and receive the data through the browser space then it will make it clumsy.

To overcome this problem we introduce a AJAX call for background process ,AJAX call overcome this problem by using the AJAX engine which handle the request and response instead of the browser.

A typical AJAX architecture looks like:



**Figure 3.3 AJAX Architecture**

There are some disadvantages to this AJAX architectural design. These disadvantages or drawbacks have been overcome by the other techniques in the project.

AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

1. Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
2. Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
3. With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
4. XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
5. AJAX is a web browser technology independent of web server software.
6. A user can continue to use the application while the client program requests information from the server in the background.
7. Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
8. Data-driven as opposed to page-driven.
9. Not easy to replace components - It is very difficult to replace any component with a better design or performance without affecting the whole architecture.

**3.3.2 LIVE USER LISTING**

The first phase of the project is the data collection from the user which is done by the game played between the human player. These human player knows each other to give the challenge request to the desired player .To make the player know each other ,the of live users should be displayed to the each individual player.

The other reason to have the live user listed is to enable the users to give request to the other player ,then the request have to be handed over to the recipient player which is done by request handler mechanism

**3.3.3 REQUEST HANDLER**

After the request is being given to a desired player it is handled by a mechanism called 'request handler' ,this mechanism adds the request to the request box the recipient player along with two buttons ,one stating accept and the other stating decline.

Both the accept and decline operation are handled by the request handler. if a player selects accept then a accept signal for the corresponding request is sent to the request handler and further actions are taken if the player clicks decline then the particular request is removed from the request queue of the particular player.

**3.3.4 GAME BOARD MANAGER**

The major task of the game board manager starts after the player click accept button and the accept signal being sent to the request handler ,after receiving the accept signal from the recipient player the request handler change the status of the both player to busy state and then transfer the control to the game board manager

The game board manager check for the status of the both players and life left for the player and then proceed with creating a instance of the game board for the player to have their game. In addition to this ,the game board manager check the game type also (that is whether it is ordinary game or opponent is a bot) so if it is bot it sets the opponent player as a bot or else it will set the opponent player as a object representing the human player.

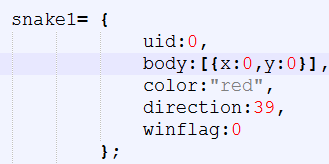
**3.3.5 DATA COLLECTOR**

The important job of the data collector is the data collection from the user through their moves. Whenever a key is pressed and AJAX call with following data is sent to the data collector module

1. JSON Object of red snake.
2. JSON Object of blue snake.
3. Head position of red snake.
4. Head position of blue snake.
5. Length of both the snake.
6. Current moving direction of both the snakes.
7. Distance between the X co-ordinates of the snake heads.
8. Distance between the Y co-ordinates of the snake heads.
9. Distance between the food and the snakes.

The greater advantage of this kind of data collection is no need for the data cleaning since every data being collected is being gathered consciously and only required data is collected. The JSON object of the snakes have the following attributes

1. User ID it belonged to.
2. Array of body parts object.
3. color of the snake.
4. current moving direction of the snake.
5. winning flag value of the player operating the snake.

Below shown is the code snippet of the JSON object 

**Figure 3.4 snake JSON Code**

**3.3.6 PREDICTOR**

The predictor module is basically used for predicting the user's next moves ,without this module the computer would be just a path finding snake because only this helps the computer snake to predict the user's next move in advance and helps it make a better move. This module is not active all the times it get activated occasionally because there is no need to predict the opponent's move always ,the need for prediction occurs only in the following situations

1. When a opponent approach the food.
2. When opponent entered the imaginary boundary of the comp snake.

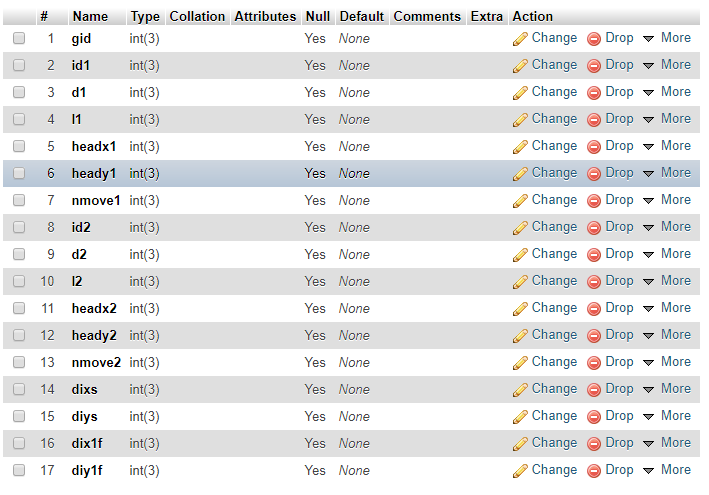
An imaginary boundary is covering the computer snake in all the four direction once the presence of the enemy snake is found inside the boundary the current game board situations are sent to the predictor module for prediction of the opponent's next move. The data sent for prediction include the following

1. JSON object of the both the snakes.
2. DIXS - difference in X co-ordinates of the both snake's head.
3. DIYS - difference in Y co-ordinates of the both snake's head.
4. DIX1F - difference in X co-ordinate of the opponent snake's head and X co-ordinate of the food.
5. DIY1F - difference in Y co-ordinate of the opponent snake's head and Y co-ordinate of the food.

**3.3.7 USER MOVE DATABASE**

The role of the user move database is to store the moves given by the users while playing the game ,this database consist of only one table ,that is next move table .This table include some of the important attributes used for prediction like dixs ,diys ,dix1f and diy1f.This table totally consist of the seventeen attributes in which some are insignificant in the current required of the project they are ,game board ID,ID of the first player and ID of the second player

The structure of the next move table is given below



**Figure 3.5 Database structure**

**CHAPTER 4**

**SYSTEM REQUIREMENT**

**4.1 HARDWARE REQUIREMENTS**

PROCESSOR : AMD A4-4300 2.5GHz

HARD DISK DRIVE : 50GB

RAM : 2GB

**4.2 SOFTWARE REQUIREMENTS**

OPERATING SYSTEM : Windows 7 and above

FRONT END : HTML, CSS3

BACK END : Java 8, Java servlet 3.0

SESSION STORAGE : Local Session storage

DATABASE : MySQL 5.7

APIs : MySQL J connector ,org.JSON ,servlet

**CHAPTER 5**

**IMPLEMENTATION MODULES**

**5.1 MODULES TO BE IMPLEMENTED**

1. LIVE USER LISTING MODULE
2. GAME CONTROLLER MODULE
3. USER SNAKE CONTROLLER MODULE
4. GAME SYNCHRONIZER MODULE
5. DATA COLLECTION MODULE
6. PREDICTION MODULE

**5.1.1 Login service**

This project is targeted at developing the unknown user gaming system in which players do not need to expose their true identity ,they can just have their own nick name for reference and can start playing with other online unknown players.

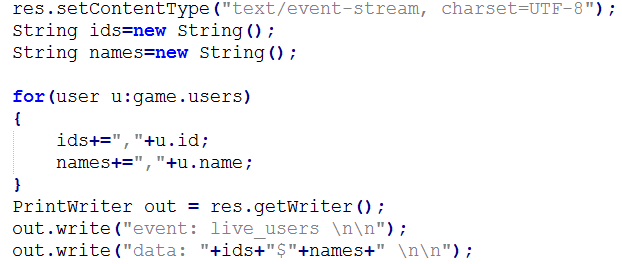
The login page is very simple ,it just consist of the getID button and a start game button in the centre of the page ,if the user clicks the start game button without getting an ID then page will give no response because the system uses the local session storage for session maintenance so the procedure is to first get a ID by clicking the getID button and then proceed with entering the home page ,on clicking the getID button the browser prompt for the nick name where he/she can enter their desired nick name and then click start game button to proceed to next page.

**5.1.2 LIVE USER LISTING MODULE**

The biggest challenge in such multiplayer gaming is connecting the people all over the network to particular system ( web application).This difficult process is done by the live user listing module.

This process is done by the SSE servlet file named "LiveUserLister.java" ,this is a simple module which has a return type as event stream ,the event name sent by the file is 'live\_users' which is then received by the client side and rendered accordingly ,for example the client own ID should be hidden from him for the avoiding the self requesting problems.

The code snippet of the live user listing is shown from the server perspective



**Figure 5.1 Code snippet live user listing**

The code snippet of the live user listing is shown from the client perspective

live user listing client.PNG

**Figure 5.2 Code snippet live user listing client**

**5.1.3 GAME CONTROLLER MODULE**

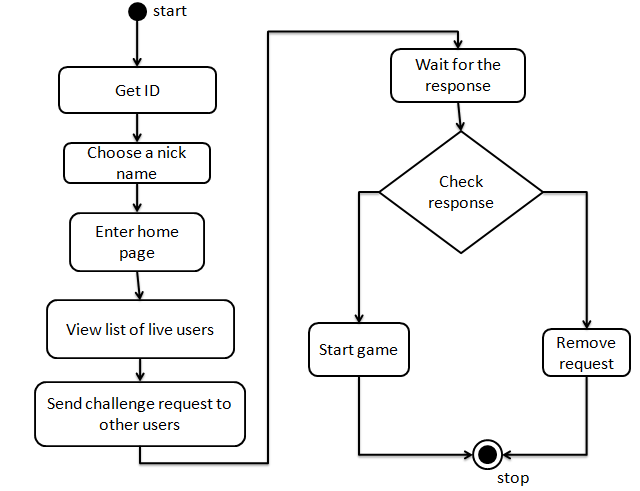
The game controller module is the starting point the game in this system. The game controller module is responsible for the listing the request sent to and received from other users ,that is comprises of the request handler mechanism ,Let us see how the request from one user is transferred to other user and how response is handled by the request handler mechanism.

First in the client side the LIVE USER LISTER MODULE takes care of listing the live users to the current player ,by seeing the list of player the user select the opponent's name with whom he/she wants to play ,then give the challenge request to the desired player just by clicking his name .This request have to be intimated tot the other player ,this is done by the request handler mechanism .On client side when he /she clicks the opponent name the following details are sent to the server

1. Opponent's ID .
2. his/her own ID
3. operation code to getID file.

On the server side the getID with operation code t=1 has the following work to do ,first check whether the recipient is a computer or a human player if human player then add the request to the recipient's request list else just return -1 as a intimation to the client agent(Browser) ,in this flow the next operation is completely depend on the recipient's ,if the recipient of the request accepts it ,then game starts else the request is removed from the recipient request list. On clicking the accept the following data is sent to the server his/her own ID ,challenger's ID ,JSON object of the both the snake .

The flow of the process is shown below



**Figure 5.3 Flow of getting ID**

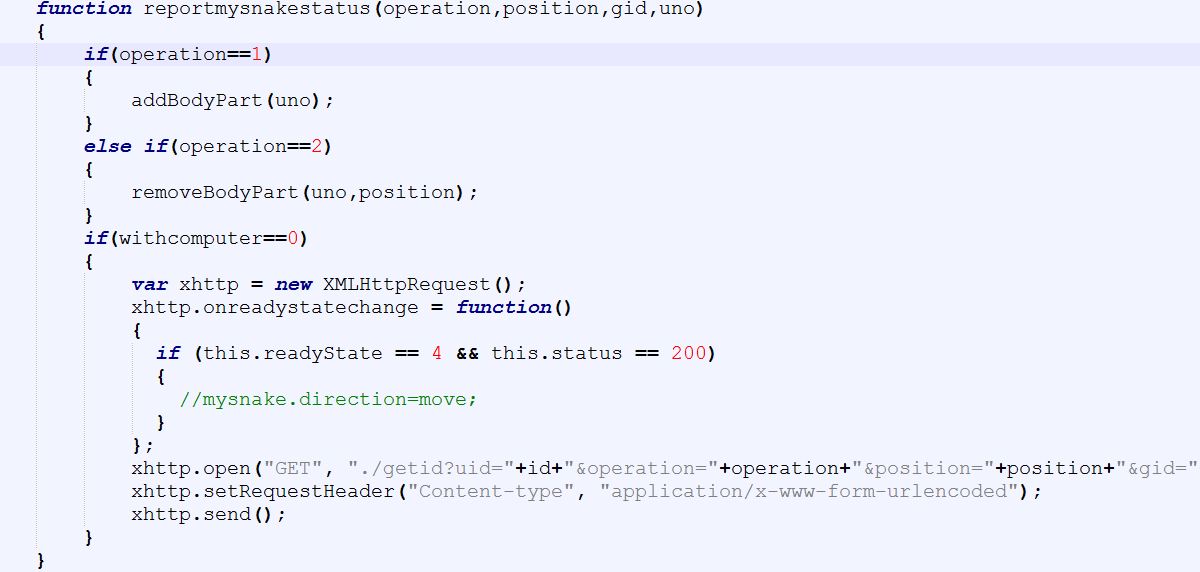
**5.1.4 USER SNAKE CONTROLLER MODULE**

The main aim of the snake controller module is the controller synchronization and allowing the user to control his snake.The user totally has four control in the game. In this game the user has four movement controls left ( ← ) ,right ( → ) ,top ( ↑ ) and down ( ↓ ). On pressing any of these button a key code is sent to a function with in turn checks the **user’s id** and report the server about local changes of the snake .The receives the report and push it to its listeners and listeners gets the announcement from the server and makes the changes to the local game board. The following changes of the local game board is reported to the server

* + 1. Snake’s position changes
    2. Addition of body part to a snake.
    3. Removal of body part to a snake.
    4. When body part count of any snake gets below THREE .

These moves are saved by the move collector module by a database code .The reason to use the data collection module is to store the user moves to use it in the future for the computer

The below code snippet shows the snake controller module in client side.



**Figure 5.4 snake control client side**

This module consist of only two sub modules they are

1. addBodyPart
2. removeBodyPart

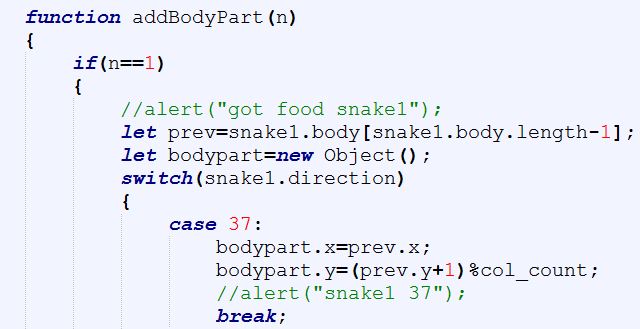
**addBodyPart**

This module is used to add the body part to the any given snake at given satisfying situations

**removeBodyPart**

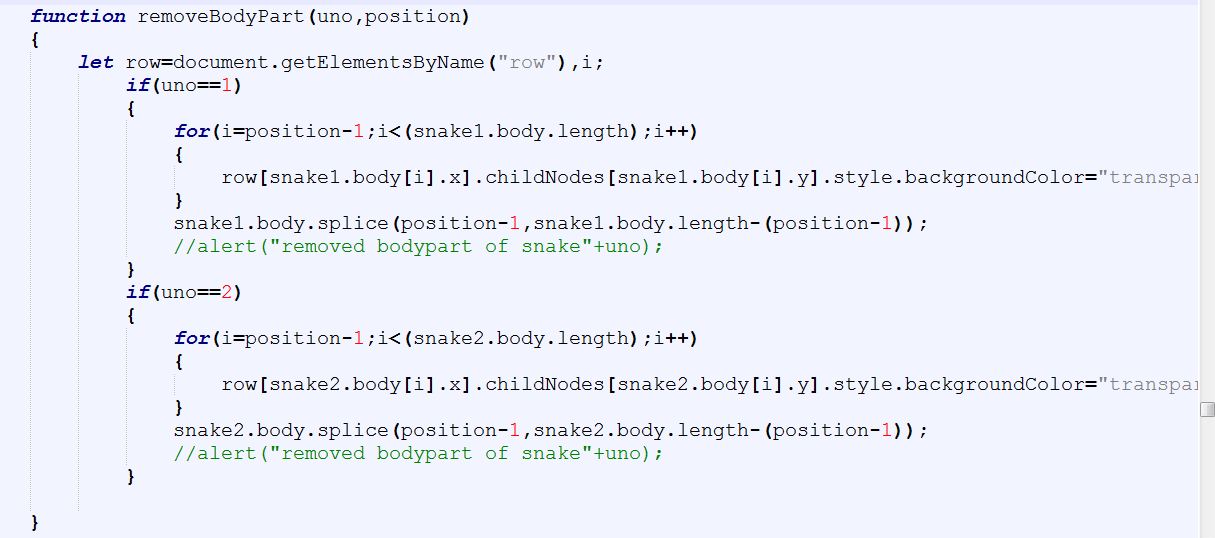
This module is used to remove the body part at given position of a given snake ,this module is triggered when a snake hits another snake’s body.

The below code snippets shows the add body parts and remove body parts



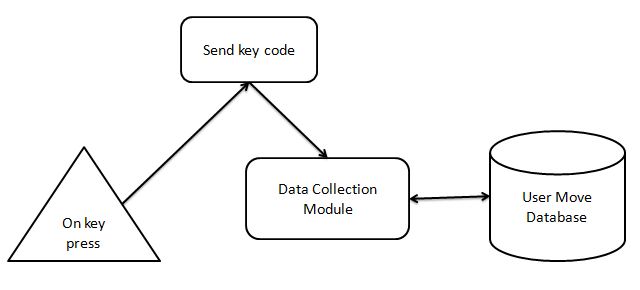
**Figure 5.5 Add body part code snippet**

Due to the shortage of space only important line are shown ,actually the add body part does the adding the body parts to the snake by checking the current direction of the snake and finding the last body part of the given snake.



**Figure 5.6 remove body parts code snippet**

The below diagram shows the flow of getting the moves and storing it for future use.



**Figure 5.7 flow for data collection**

**5.1.5 GAME SYNCHRONIZATION MODULE**

Another important module in the game is the synchronization module, because the success of the prediction purely depends on the cleanliness of the data being produced In order to produce a good dataset a better synchronised game model is required ,the first synchronised model proposed perfectly matched the requirements of the system ,but it has compromised the efficiency of the system ,so the project needed a alternative synchronised model

FIRST MODEL :

This model used the server oriented system where every synchronisation process happens in the server side and the client just acts as an display for the users. Because of this server oriented model the proposed system failed to serve the better gaming experience the reason for that is server have to handle lots of data per second and also send it back to the client

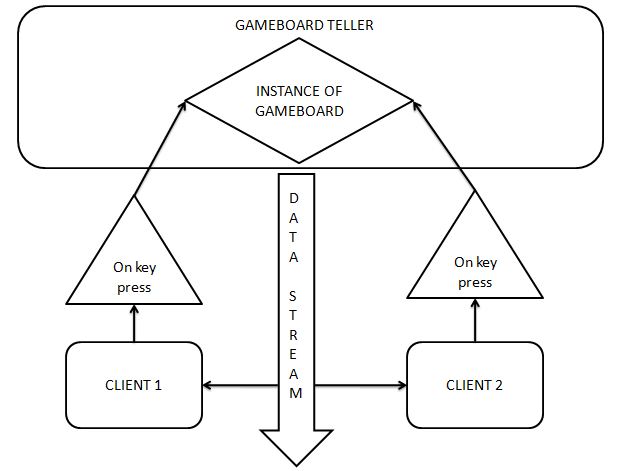
SECOND MODEL :

In this second model the entire is localized to the client agent called browser for faster game play. This second model is pure depend on the performance of the client agent (browser) ,since the instance of the game board created separately for two different clients. In this model there are three synchronization should be taken care of, they are

1. Snake position synchronization

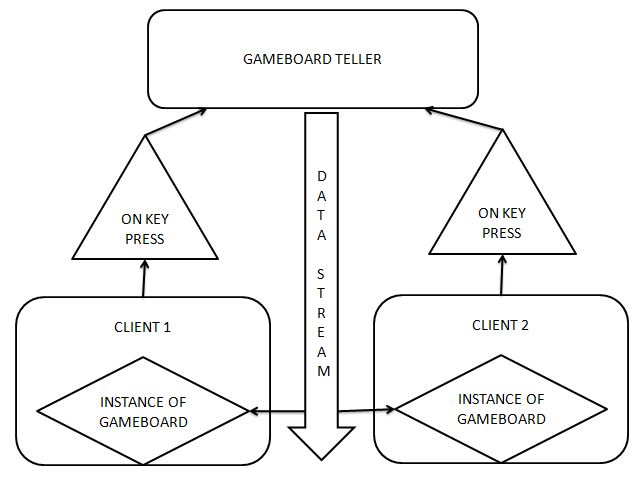
2. Body part change synchronization

3. Food Synchronization

****

**Figure 5.8 model 1 for synchronization**

The another important synchronisation to be done is timestamp calibration, which is depicted in the following diagram. Every synchronization happens by flags . Both Snake position and body parts are synchronized by flag values and event stream sent by the server. But the food Synchronization is not implement using flag values it is fully dependent on the start time calibration ,incase of the timestamp not synchronized then the game will be paused for both and will be started again once timestamp gets synchronized.



**Figure 5.9 model 2 for synchronization**



**Figure 5.10 Timestamp calibration**

The above picture show the example of the timestamp calibration first the user 1 seek the data at 50th millisecond and the server tells the user1 to meet the server after 110 milliseconds that is at 160th millisecond(50+100=160) and second user seeks data at 80th millisecond so the server tells him to meet the server after 80 milliseconds that is 160th millisecond if the second user or the first user fail to meet the server at the given time then this calibration continues ,if they met the server at given time then the timestamp difference will be zero so the game start signal will be sent to both clients.

* + 1. **PREDICTION MODULE**

This is the another important module of the system since it is

Responsible for the next move prediction of the opponent in critical situations. In this project this is done by a classification algorithm called naive bayes classifier.

In machine learning, **naïve Bayes classifiers** are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features.Naïve Bayes has been studied extensively since the 1960s. It was introduced (though not under that name) into the text retrieval community in the early 1960s and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines It also finds application in automatic medical diagnosis.

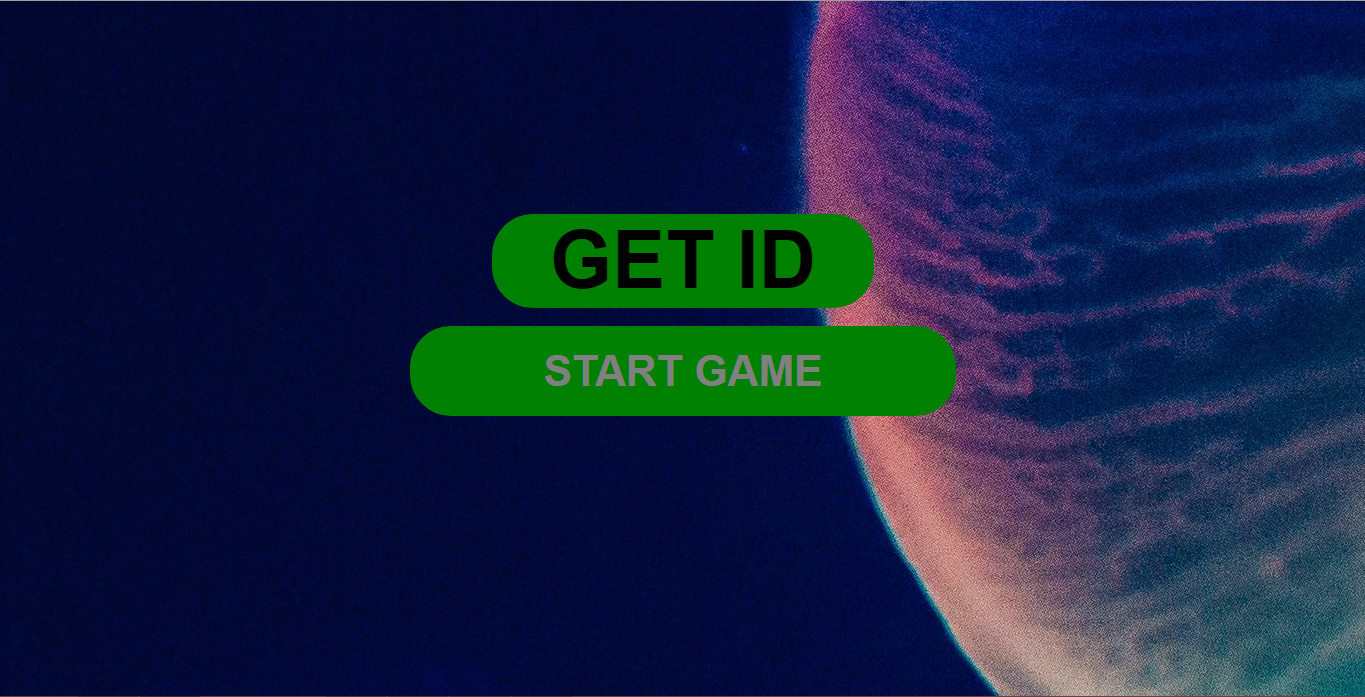
Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

In the statistics and computer science literature, naive Bayes models are known under a variety of names, including **simple Bayes** and **independence Bayes**.All these names reference the use of Bayes' theorem in the classifier's decision rule, but naïve Bayes is not (necessarily) a Bayesian method.

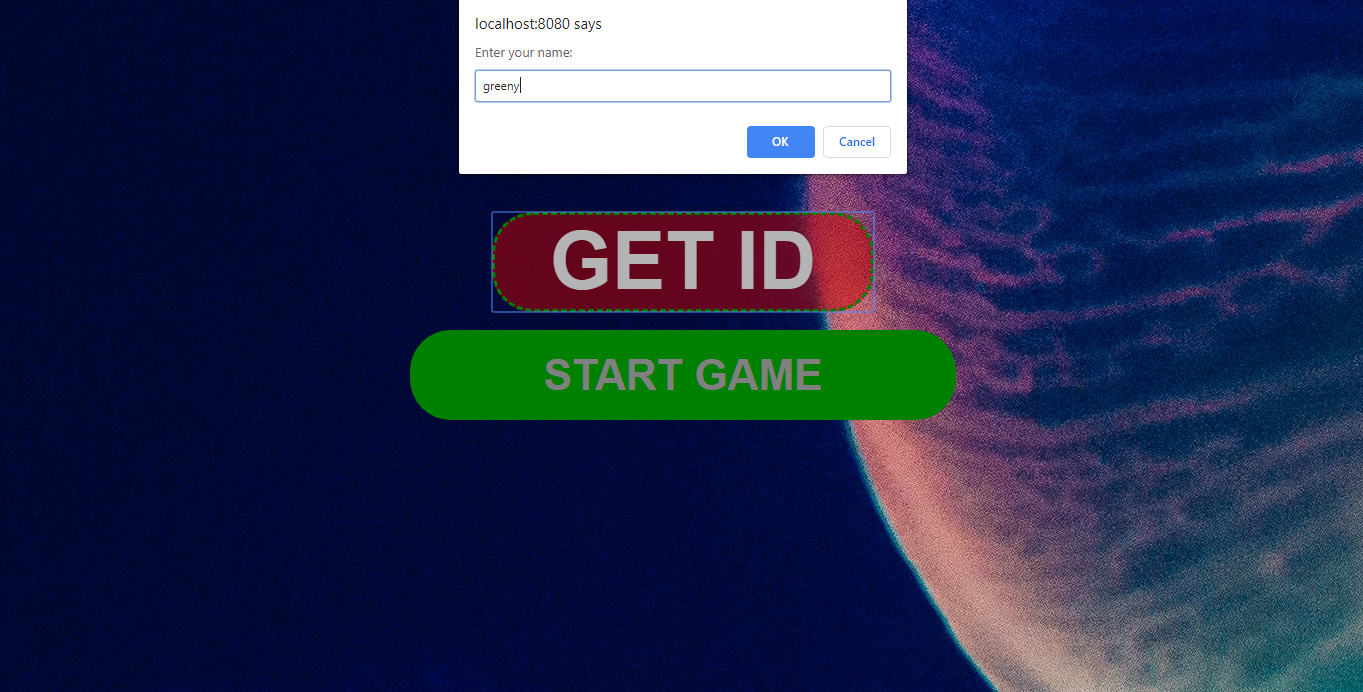
**CHAPTER 6**

**SNAPSHOT OF MODULES**

**6.1 LOGIN PAGE**

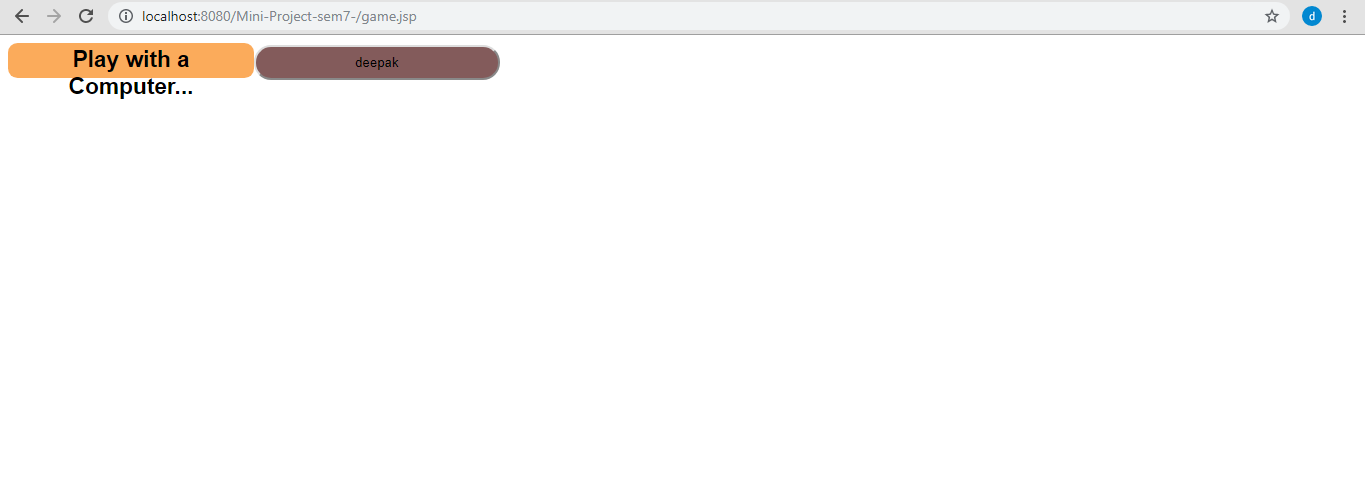
****

**Figure 6.1 Login Page**

**6.2 GETTING ID** 

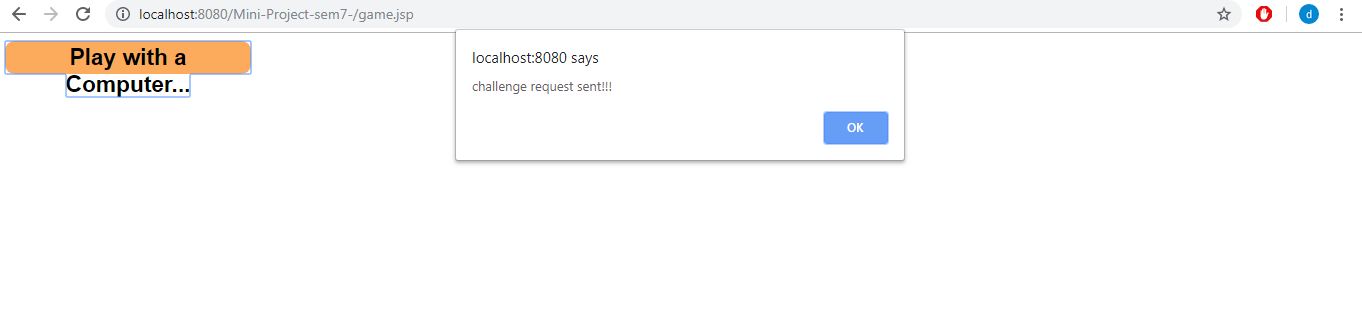
**Figure 6.2 Getting the ID**

**6.3 GETTING LIVE USERS**



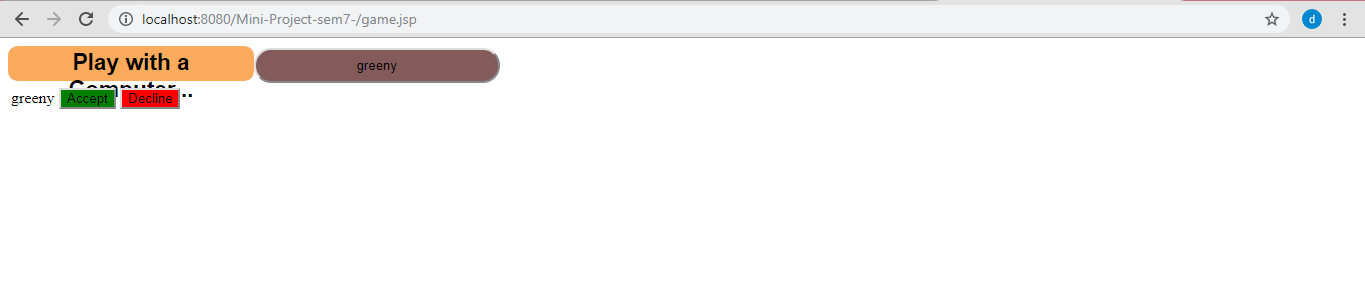
**Figure 6.3 Getting Live Users**

**6.4 SENDING REQUEST**

****

**Figure 6.4 Sending request**

**6.5 RECIEVING REQUEST**

****

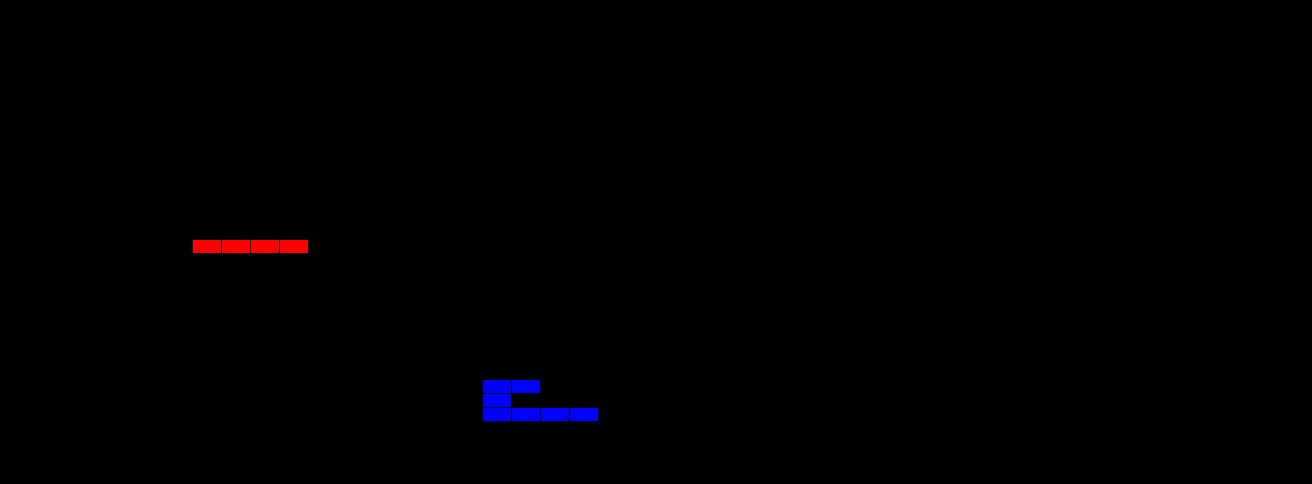
**Figure 6.5 Receiving request**

**6.6 CALIBRATING GAME**

****

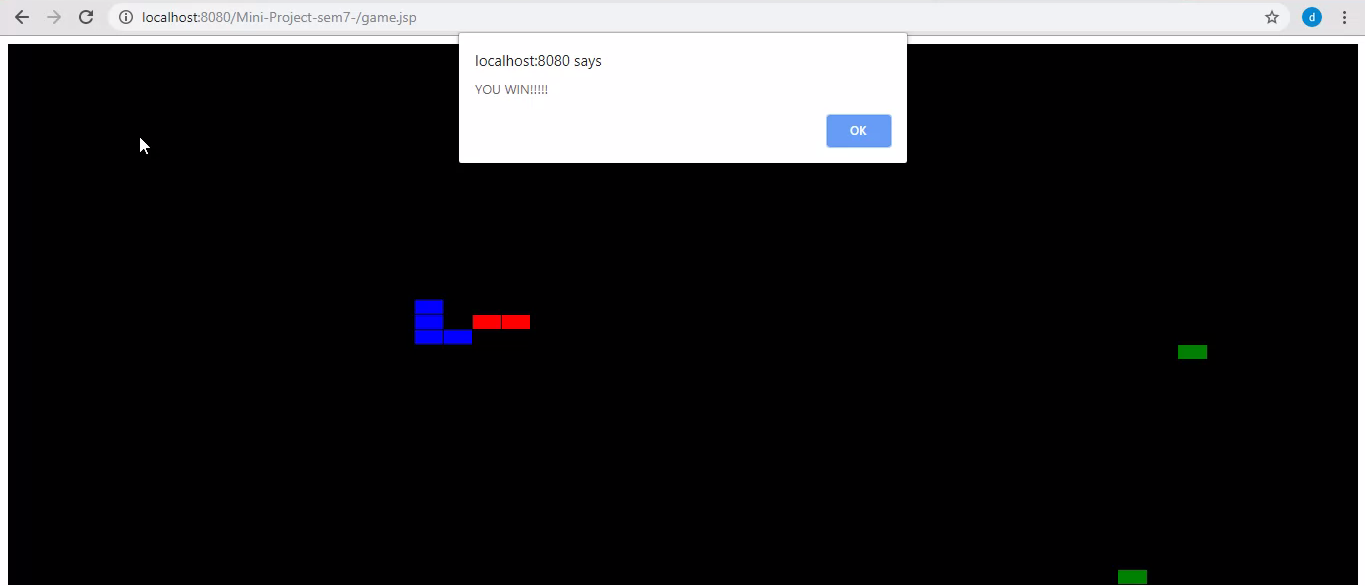
**Figure 6.6 Timestamp calibration screen**

**6.7 STARTING GAME**

****

**Figure 6.7 Starting the game**

**6.8 ENDING THE GAME**

****

**Figure 6.8 Game End**

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

Thus a game with could be played in multiplayer mode from separate devices is developed. Collected the user moves data. Did a prediction on collected data for helping the computer snake to have a better moves again the human a opponent.

The future scope is as follows:

1. Construct a more efficient model to predict a more personalized user move.
2. Enabling more than two players to play on a single game board.
3. Enabling the dynamic game board size.

.

**REFERENCES**

1. <https://github.com/topics/slitherio>
2. <https://github.com/huytd/agar.io-clone>
3. <https://github.com/rafaelsofizadeh/modcon-io-game>
4. <https://github.com/vzhou842/example-.io-game>
5. <https://github.com/magicsea/h5game>
6. <https://github.com/fedofcoders/tienk.io>