



12/10/2021

Game Website - Final Submission

CST2120 Web Applications and Databases

Avinesh Culloo (M00776456)

BSc Computer Science (System Engineering)

Table of Contents

Introduction.....	2
How does the game work?	2
Additional game functionalities.....	4
Registration Functionality.....	5
Login functionalities	7
Ranking functionalities.....	8
Other functionalities	8
Difficulties encountered during development.....	10
Game Screenshots	10
1. Game Loading page	10
2. Beginner Level.....	11
3. Normal Level.....	11
4. Time Attack Level.....	12
5. Game Over	12
Storage Screenshots	13
1. Local Storage	13
2. Session Storage	14
Special Thanks to.....	14
Conclusion.....	14

Introduction

The game, Naagin has been developed in a tidy and easy to read manner making use of HTML, JavaScript, PHP and CSS. The game consists of the following 3 different levels:

1) Beginner Level

The player should not make the snakehead touch any part of the body, else the game will be over. Moreover, the snake can pass through walls and the snake will come out of the other side. For each food the snake eats, the score will increase by 10.

2) Normal Level

The player should not make the snakehead touch any part of the body and also, the wall. Else, the game will be over. In addition, to increase the difficulty level, the speed at which the snake is moving will be increasing depending on the score.

3) Time Attack Level

The rules for time attack is the same as the normal level. The only difference is that there will be no increasing speed depending on the score and also, this is a 30 seconds play. When the time left is 10 secs, 'time left' will blink. Once the time is up, the game will be over.

How does the game work?

For the game development, advanced JavaScript was used to have a more efficient game.

The game was developed in the following order:

1. Display the playing board and a still snake

The paying board or the canvas is the area that will allow the snake to navigate to eat the randomly generated foods. The canvas is a 2D context, meaning that it will be drawn into a 2D space. For the snake, an initial location has been declared and initialised which marks the starting position of the snake. Then, the snake is displayed on the canvas.

2. Movement of the snake

The snake moves one step at a rate of 10px. For the snake to move to the right, we have to increase the x-coordinates of every snake's parts by 10px and for the left, the x-coordinate of every part of the snake is decreased by 10px.

Similarly, for vertical movement, the y-coordinates have to be altered. For the snake to move up, the y-coordinates of the snake's parts will have to decrease by 10px and increase it by 10px to move the snake down.

The automatic movement of the snake depends on calling the function repeatedly. To have a delayed movement of the snake instead of a rapid movement from one point to another, the `setTimeout` is used. The milliseconds of the `setTimeout` allow setting the speed of the snake.

3. Use arrows keys and letters to change the snake's direction
To detect key pressed, `addEventListener` is used on the document and will subsequently call a function named `navigateSnake`.
`navigateSnake` will retrieve the keypress code and carry out some validations in which directions to move the snake. In addition, to avoid the snake from reversing, the need to check if the snake is moving in the opposite direction of the new, intended direction. For example, if the snake is moving to the left when pressing the arrow left or A, nothing will happen.
4. Check for boundary
Other levels than beginner level, when the snake touches the wall, the game is over. For every step the snake takes, we need to verify to carry out the following conditions:
 - a) Top wall hit
If the next intended direction is less than zero.
 - b) Bottom wall hit
If the next intended direction is greater than the game window height.
 - c) Left wall hit
If the next intended direction is less than zero.
 - d) Right wall hit
If the next intended direction is greater than the game window width.

This check is carried out in a loop until the game is over.
5. Generate food locations
The purpose of Naagin is for a user to eat as much as possible to reach the top of the rank board.
The food location is generated randomly using the function `generateFoodLocationRandom`. A further check is made to make sure that the food location is not the same as the snake's location. If it is, a new food location is generated.
6. The snake's growth
Whenever the snake's head is the same as the food location, instead of popping the last element of the snake, a new body part is added to increase the snake's length.
7. Score
Whenever the snake eats, that is, the snakehead is the same as the food location, the score increases by 10. Increasing speed at the normal level will increase the snake movement speed.

Additional game functionalities

1. Blinking score

When the current score is higher than the high score, “your score” changes from white to “new high score” with blinking effects.

2. Classes

To make the game more efficient, classes were introduced in the Naagin development. The different classes are as follows:

i. Canvas class

The Canvas class draws the snake, food, canvas on the game window.

ii. Food class

The Food class calculates the coordinates for the food location, calls the class Canvas to draw the food as per the generated location and checks if the snake ate the food.

iii. Snake class

The Snake class calls the Canvas class to draw the snake, calculates the next position of the head, calls the Food class to check if the snake has eaten the food, whether the snake has touched the wall or not and finally, whether it has eaten its own body.

3. Blinking time left

The time attack level is a 30 seconds play. When the time left is less than 10 seconds, “Time left” and the number of seconds left start blinking.

4. Sound effects

Different sounds are played on different occasions

a) Eat sound

When the snake eats the food, the eating sound is played.

b) Bump sound

When the snake collides with the wall, the bump sound is played.

c) Dead sound

When the snake eats itself, the dead sound is played.

5. Counter

Before the game starts and after the level choice is made, there is a five seconds countdown that allows the user to read the rules of the chosen levels.

6. Animated snake

When the snakes move inside the game window, it changes colour and its head is always red to differentiate the head amongst the other body parts.

7. Increasing speed

For the normal level, to make it more interesting, with an increasing score, the snake movement speed will also increase.

8. Left-hand people

Left-hand people are also taken into consideration. For these people, they can make use of W, A, S, D to move the snake around.

9. Pause

The game can be paused either by clicking on the button pause or by pressing P on the keyboard. This option is only available for beginner and normal levels.

Registration Functionality

For a player to enjoy Naagin, he/she will have to register through a registration form.

1. Username Validation

a. Length check

To make sure that the username input field is not empty, the length of the value of the username input field is calculated and if it is zero, “*required” is displayed.

b. Username structure

A further check is made to make sure that the username is 8 to 20 characters, no _ or . in the beginning, no __ or _ or . or .. inside the username and no _ or . at the end of the username.

c. Existing username

A further check is made to make sure that the username entered does not already exist.

2. Email Validation

a. Length check

To make sure that the email input field is not empty, the length of the value of the email input field is calculated and if it is zero, “*required” is displayed.

b. Email structure

A further check is made to make sure that the email is in the following format: name@domain.com/or any 2 digits characters.

3. Password Validation

By default, the entered password is hidden. To view the password, the player can press the eye beside the input box.

a. Length check

To make sure that the password input field is not empty, the length of the value of the password input field is calculated and if it is zero, “*required” is displayed.

b. Password structure

A further check is made to make sure that the password abides by the following conditions:

i. Min 6 elements

- ii. Containing at least:
 - a. A symbol (!@#\$\$%^&*)
 - b. Upper- and lower-case letter
 - c. A number
- 4. Confirm Password Validation
 - a. Length check

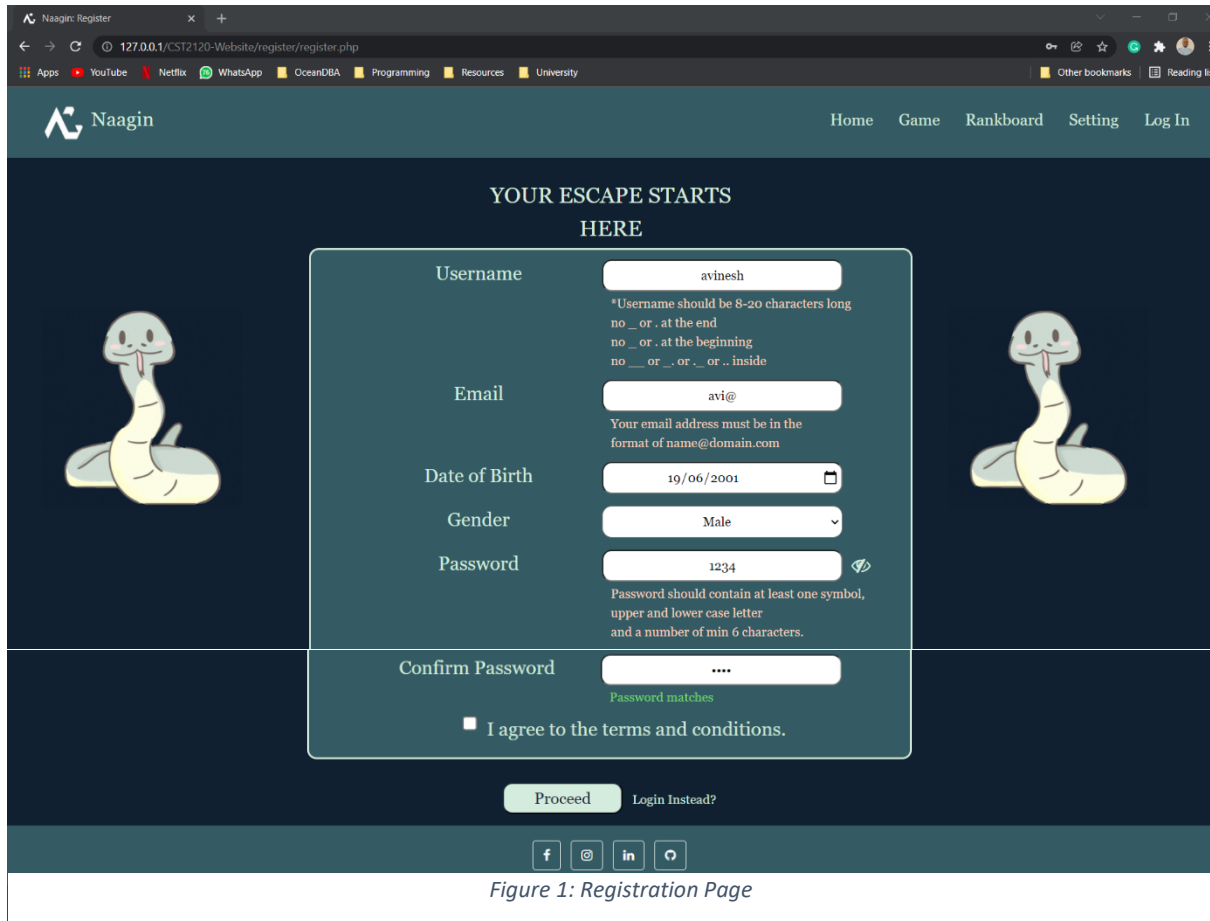
To make sure that the password input field is not empty, the length of the value of the password input field is calculated and if it is zero, “*required” is displayed.
 - b. Password comparison

A further check is made to make sure that the confirmed password matches the previously entered password
- 5. Check box

A check is made to ensure that the terms and conditions are accepted by ticking the checkbox.
- 6. Date of Birth validation

Every date after the current date is disabled.

After the form has been validated making use of JavaScript, the data is appended to the existing user object and then converted into JSON for HTML local storage.



The screenshot shows a web browser window with the URL `127.0.0.1/CST2120-Website/register/register.php`. The page has a dark blue header with the 'Naagin' logo and navigation links: Home, Game, Rankboard, Setting, and Log In. The main content area has a dark blue background with the text 'YOUR ESCAPE STARTS HERE' at the top. Below this is a registration form with a light blue border. The form contains the following fields and elements:

- Username:** Input field with 'avinesh'. Below it, a message: '*Username should be 8-20 characters long no _ or . at the end no _ or . at the beginning no _ or . or _ or .. inside'.
- Email:** Input field with 'avi@'. Below it, a message: 'Your email address must be in the format of name@domain.com'.
- Date of Birth:** Input field with '19/06/2001' and a calendar icon.
- Gender:** Dropdown menu with 'Male' selected.
- Password:** Input field with '1234' and a strength indicator icon. Below it, a message: 'Password should contain at least one symbol, upper and lower case letter and a number of min 6 characters.'
- Confirm Password:** Input field with '....'. Below it, a green message: 'Password matches'.
- Terms and Conditions:** A checkbox followed by the text 'I agree to the terms and conditions.'

At the bottom of the form are two buttons: 'Proceed' and 'Login Instead?'. Below the form is a footer with social media icons for Facebook, Instagram, LinkedIn, and Twitter.

Figure 1: Registration Page

Login functionalities

1. Username validation.

To make sure that the username input field is not empty, the length of the value of the username input field is calculated and if it is zero, “*required” is displayed.

2. Password validation.

To make sure that the password input field is not empty, the length of the value of the password input field is calculated and if it is zero, “*required” is displayed.

3. Login form validation

A further check is made to make sure that the username and password match the data stored in the HTML local storage. If it does not match, “*username/password incorrect” is displayed.

4. Navigation bar login and logout

When the user is logged in, “log out” is displayed and when it is clicked, it changes to “log in”. Once the user is logged out, the session storage is clear.

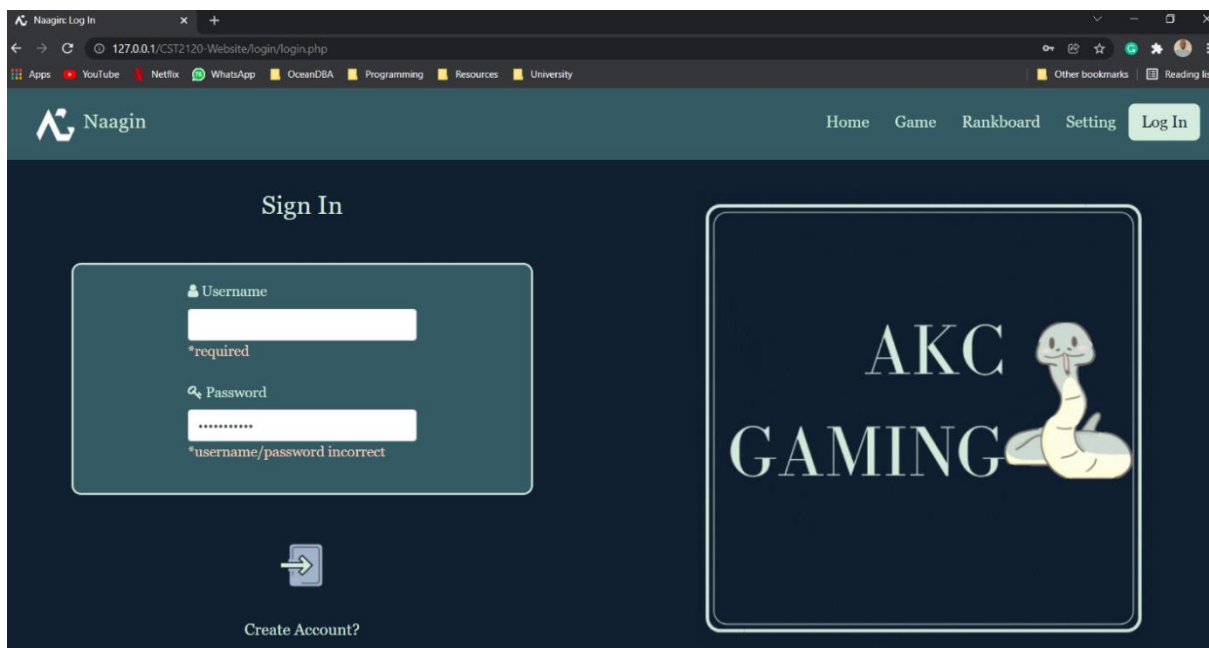
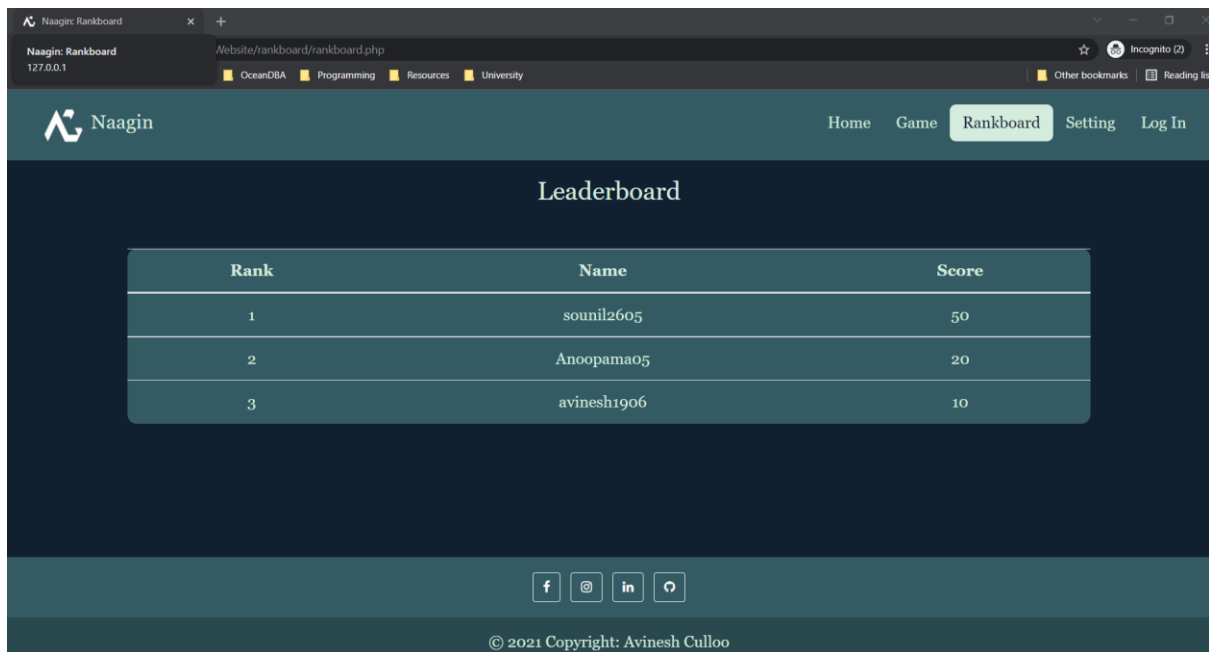


Figure 2: Log in Page

Ranking functionalities

Once the game is over, a check is made to determine the logged user's name. Then, accordingly, the score is compared with its current high score. If the new score is higher than the high score, the score is updated. Furthermore, the score is sorted to be displayed in descending order on the leaderboard page. If a user has just created an account, its score will be zero(0) by default.



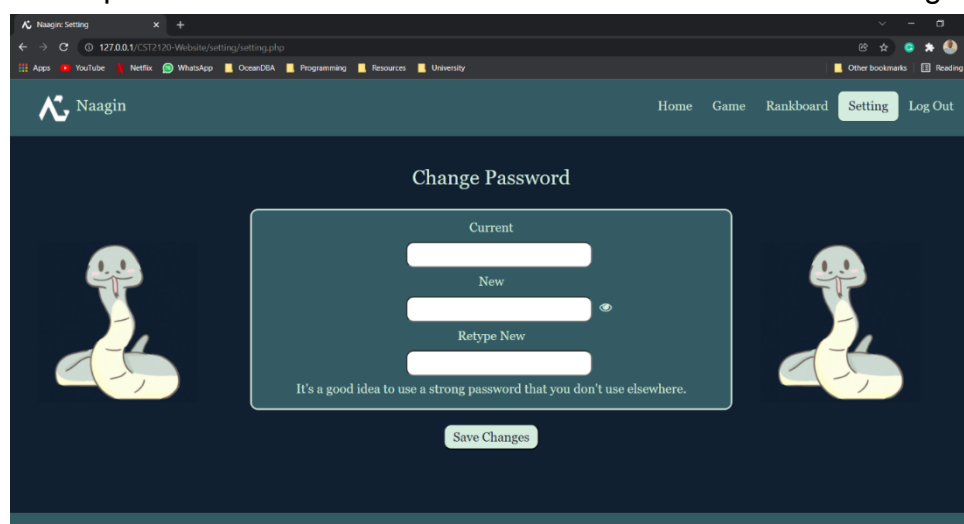
Rank	Name	Score
1	sounil2605	50
2	Anoopama05	20
3	avinesh1906	10

Figure 3: Leaderboard Page

Other functionalities

1. Setting

The logged user will be able to change its current password on the setting page. The user will have to enter its current password followed by a new password and finally, he/she will have to retype the new password for confirmation. JavaScript is used to validate the setting form.



Change Password

Current

New

Retype New

It's a good idea to use a strong password that you don't use elsewhere.

Save Changes

Figure 4: Setting Page

2. Verify Login

For a player to get access to the game and setting page, he/she will have to log in first. JavaScript is used to verify if the session storage contains any logged user. If there is no logged user, an appropriate message is prompt for the user to log in or create an account.

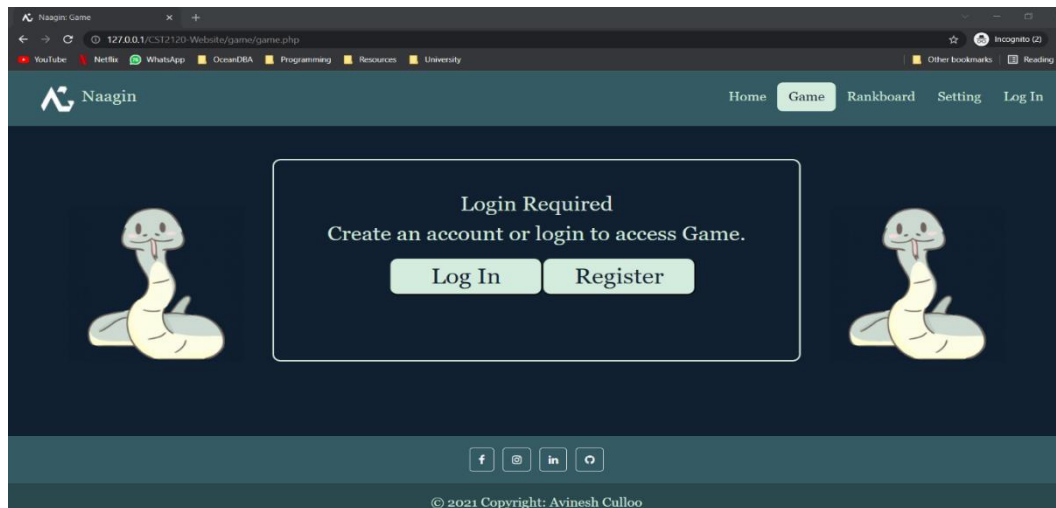


Figure 5: Game on navigation bar pressed without logged user

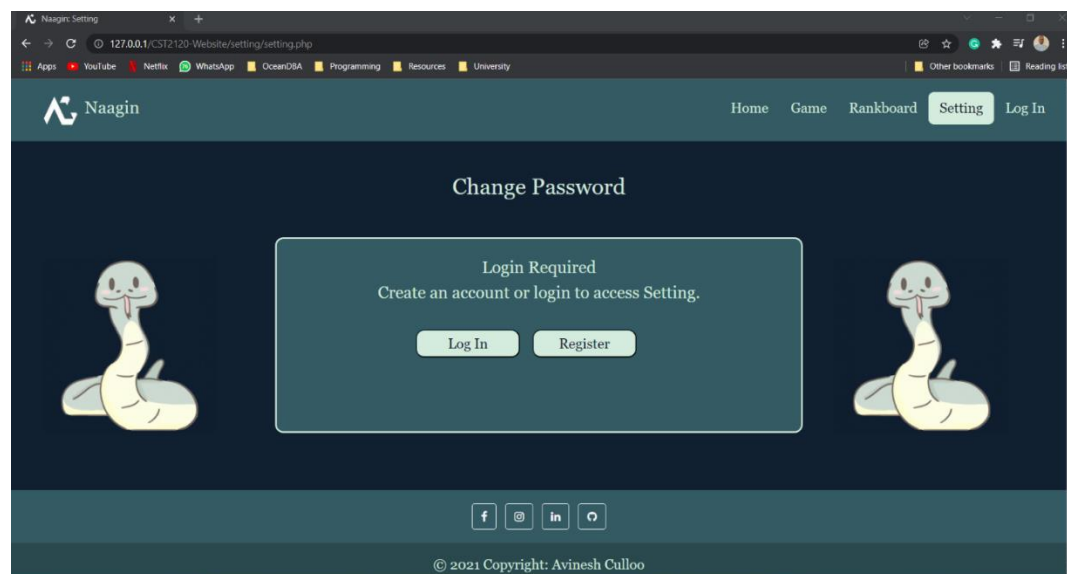


Figure 6: Setting on navigation bar pressed without logged user

3. Coloured message

To differentiate between errors and success when filling up any form, colours are used.

Difficulties encountered during development

1. CSS problem

The choice of colour and alignment of the div was time-consuming. Sometimes, the position of the div is not as expected, further research had to be carried out to have the desired outcome.

2. HTML local storage (JSON)

During the development of the registration page, once the form was submitted, the contents of JSON was deleted. After a thorough debugging, the issue was because of the form tag. I had replaced the form tag with div.

3. Game development

During the game development, I encountered the following bugs:

a. Food location generated same as the snake's location

The random food generated was the same as the snake location. Therefore, before displaying the food on the game window, further check has to be made to make sure that both locations are not the same. If it is, a new random food location is generated.

b. Time left for time attack level

The time left was not decreasing by more than 1 second. To solve this issue, the time left function has to be run in parallel with the time attack game. Once the time left is over, the gameOver function is called.

Game Screenshots

1. Game Loading page

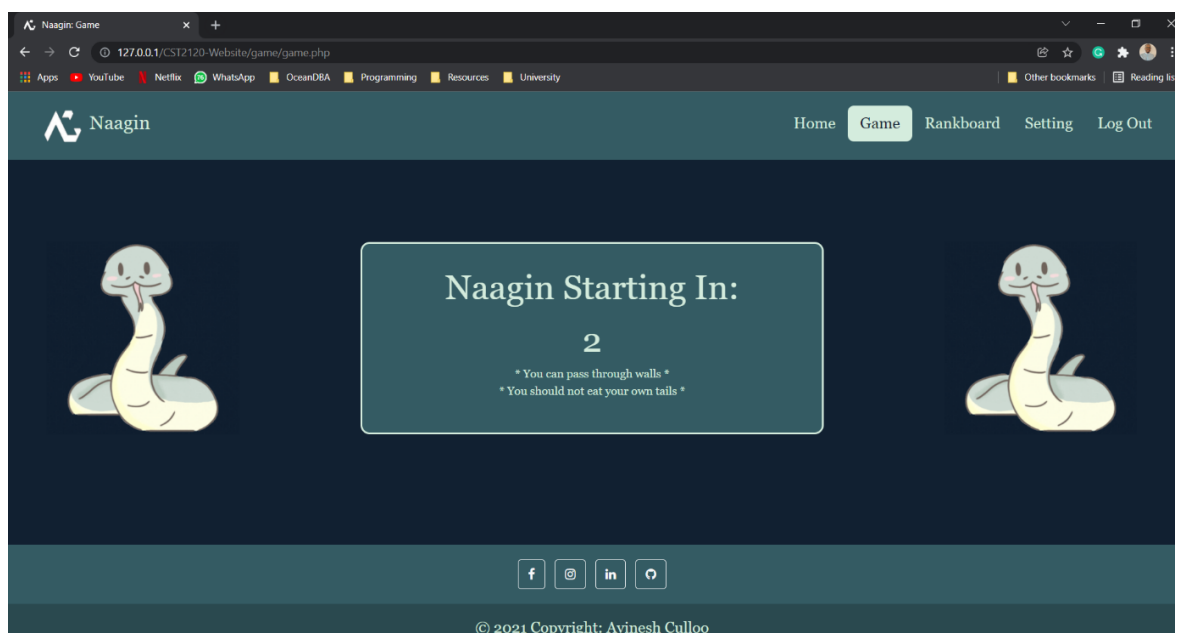


Figure 7: Game Loading page

2. Beginner Level

Figure 8 is showing that the snake can pass through the wall and it will come out from the other side.

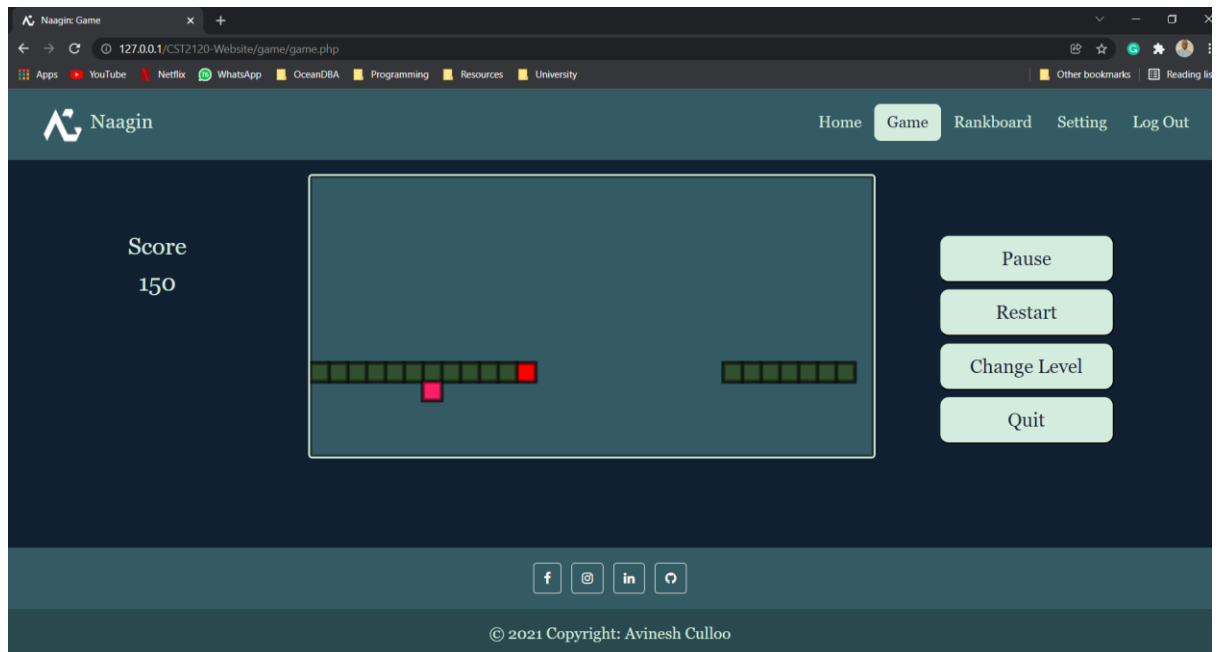


Figure 8: Beginner Level

3. Normal Level

Figure 9 is showing the effect of having a new high score.

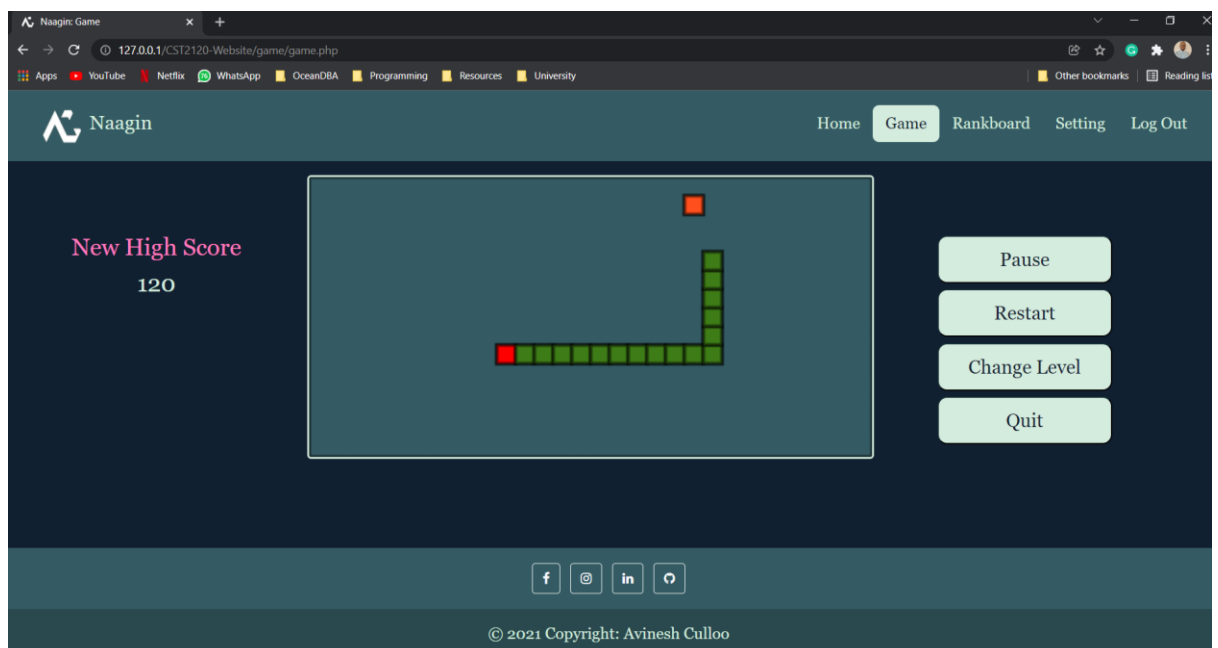


Figure 9: Normal Level

4. Time Attack Level

Figure 10 is showing the effect of having less than ten seconds left.

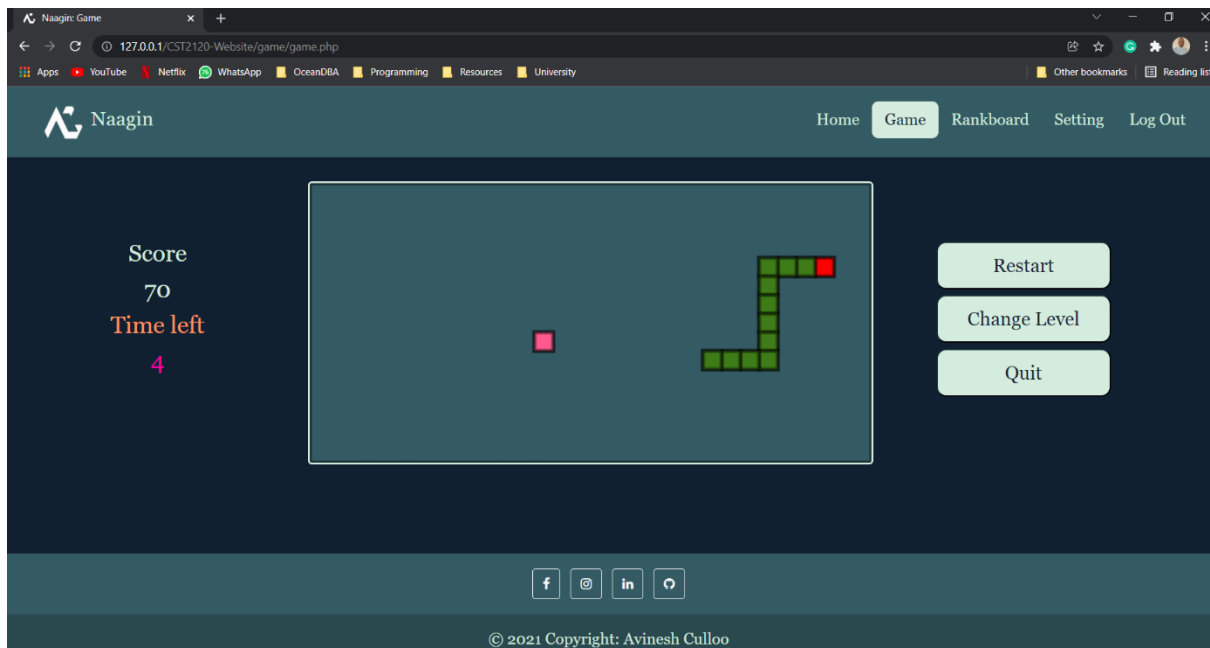


Figure 10: Time Attack

5. Game Over

Figure 11 shows the game-over window with no new high score.

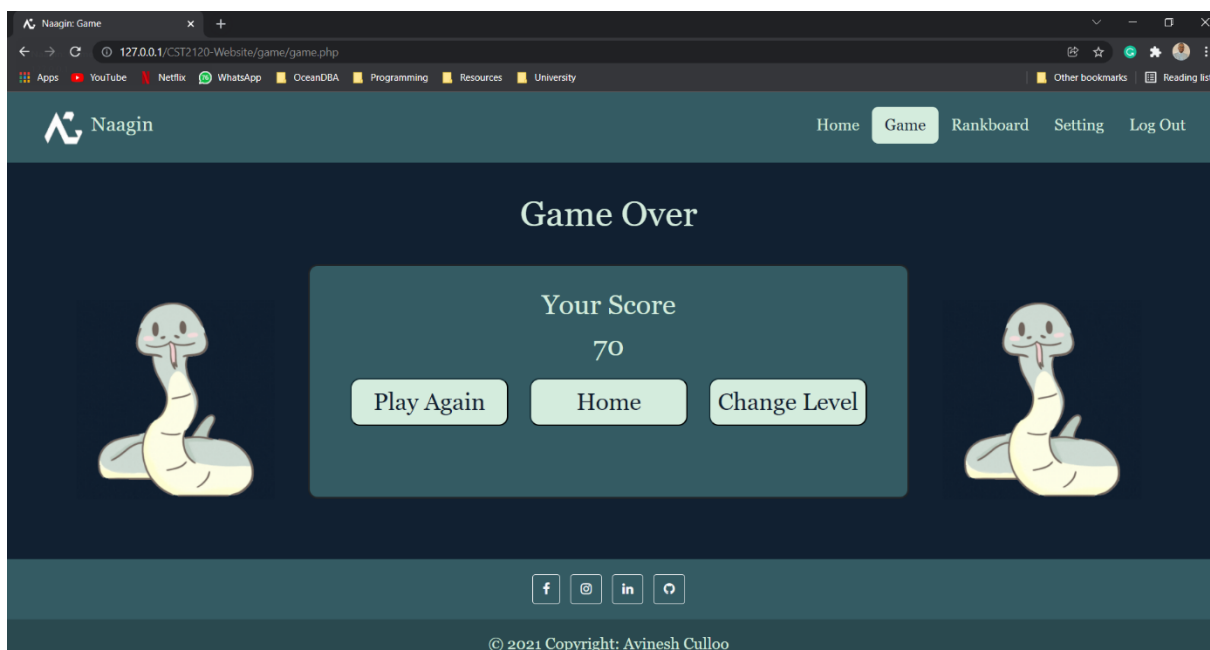


Figure 11: No new high score

Figure 12 shows a player having a new high score.

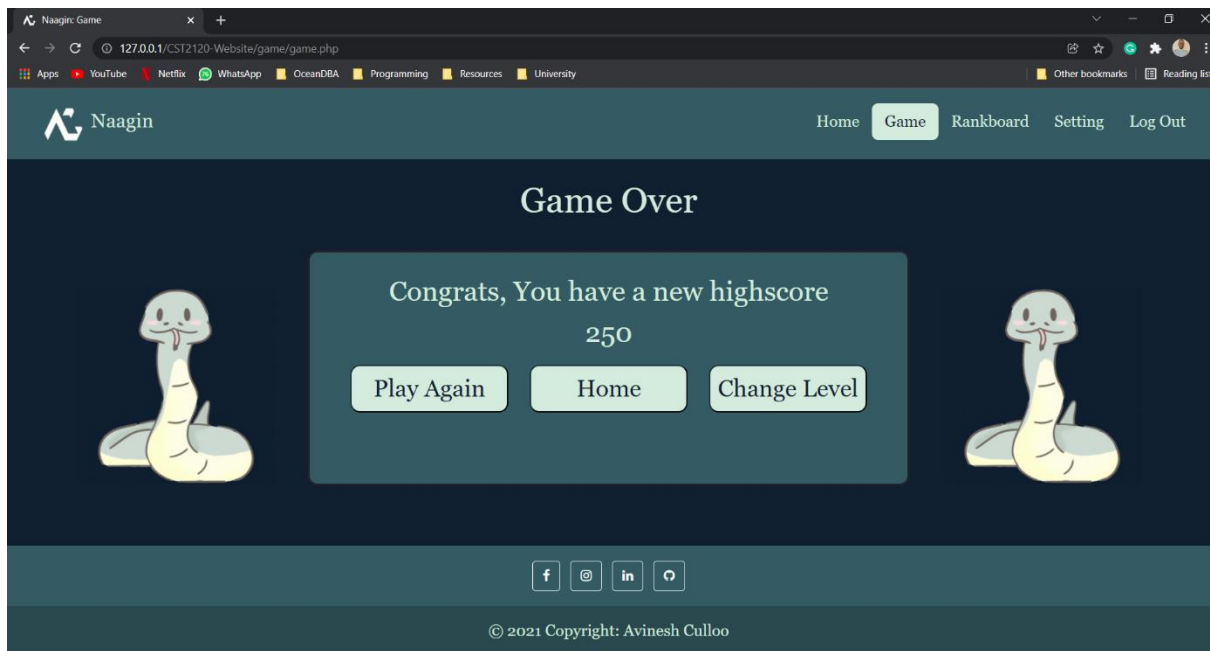


Figure 12: New high score

Storage Screenshots

1. Local Storage

Figure 13 below shows the local storage containing all the details of registered users.

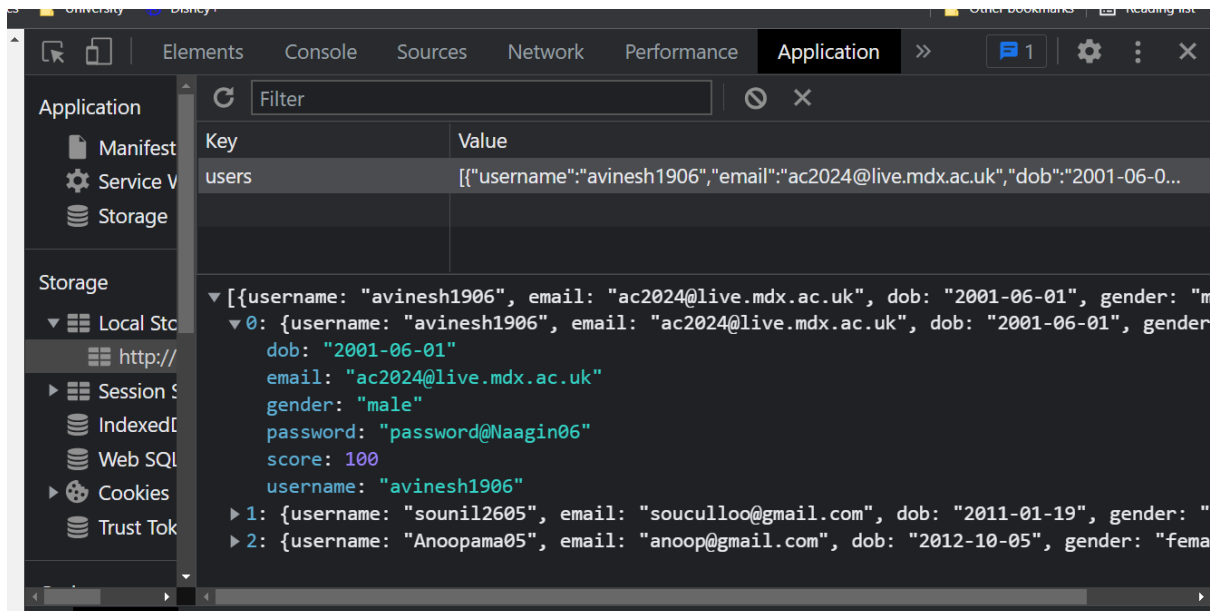


Figure 13: Local Storage

2. Session Storage

Figure 14 below shows the session storage containing the username and high score of the currently logged user.

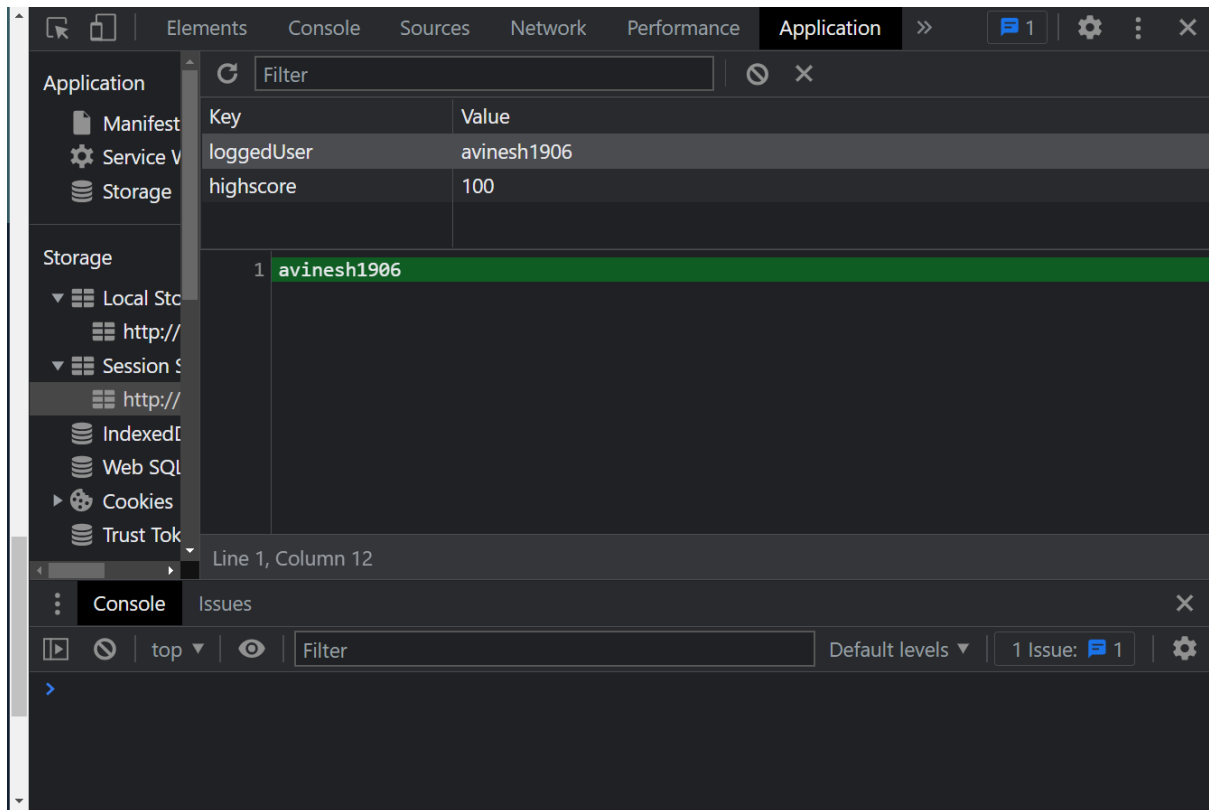


Figure 14: Session storage

Special Thanks to

Ms Ameerah Assotally, lecturer in IT and Programme coordinator for BSc Cyber Security and Digital Forensics.

Conclusion

The development of Naagin has been successful thanks to the continuous help of the lecturer, the lecture slides and recorded videos on myUniHub.