Snake Game Use Case

Document Information

|  |  |
| --- | --- |
| **Document Title** | Snake Game Use Case |
| **Document Owner** | Marvel Okafor |
| **Week** | 12 |
| **Version** | 1 |
| **Status** | Completed |
| **Date** | 03/22/2022 |

1. Brief Description

The Snake Game starts when the index.jsp page is loaded for the client. The client is given the option to either sign up or login. If they don’t have an account then they have to create one then sign in. After signing in, they are taken to the snake\_game.html page where they can play the snake game. After playing, they can view the top 10 high scores. The Program ends when the user clicks the sign out button or when the snake dies in the game.

1. Actors

* Customer/User
* MySQL Database

1. Pre-Conditions

* If accessing the game for the first time the customer has to signup/create an account.
* The customer has to login to access the game
* The customer has to use a desktop computer to play the game.
* The customer needs to have internet access to play the game.
* The front-facing app/servlet needs to connect to the database.

1. Basic Flow
2. The program starts and displays the index.jsp page for the user. The user has two options: Login or Signup.
3. If the user chooses the sign-up options, they can create an account with values such as first name, last name, username, password with the sign-up options.
4. The receiveSignup Servlet receives the user’s information and calls the ProcessUsers class.
5. The ProcessUsers class Connects to the Database and adds the User’s information to the database.
6. The receiveSignup Servlet tells the user of the success and asks the user to login in the index.jsp page.
7. If the user chooses the sign in option, then the User puts their login information and click login.
8. The receiveLogin Servlet receives the login information and calls the ProcessUsers class.
9. The ProcessUsers class connects to the database and grants access to the user.
10. The ProcessUsers class opens the Servlet\_Snake\_Game servlet page for the user.
11. The User interacts with the Servlet\_Snake\_Game servlet page by playing the snake game, signing out, or viewing high scores.
12. If the User chooses to play the game, they just need to press any of the arrow keys and the game starts. They keep playing until the snake eats itself or hits the wall.
13. If the User chooses to sign out, their session is ended and they are taken back to the index.jsp page.
14. If the User chooses to View High Scores, then they are taken to the Servlet\_High\_Score servlet page that displays the Top 10 high scores, a sign out button and a go back to game button.
15. If the user clicks the sign out button, they are taken to the index.jsp page.
16. If the user clicks the game button, then they are taken back to the Servlet\_Snake\_Game where they can continue to play the snake game.
17. Alternate/Exception Flows

2a – Before the index.jsp page sends the account creation information; it checks to see if the passwords are matching. It also makes the username and first name fields required. This means that the page won’t go anywhere until the fields are filled up.

4a – When the ProcessUsers class connects to the database, it checks username and first name the user has provided against information in the database using hibernate.

4b – If the username already exists in the database, then the ProcessUsers class returns a message to the receiveSIgnup Servlet that returns the message in an alert prompt to the user that the username has already been taken.

4c – If the First name already exists in the database, then the ProcessUsers class returns a message to the receiveSIgnup Servlet that returns the message in an alert prompt to the user that the first name has already been taken.

4d – If the first name and username are unique, then step 4 in the basic flow executes.

8a – Before the ProcessUsers class grants access to the user, it first checks the username and the password the user has provided.

8b – To check the username, the ProcessUsers calls the checkUsername method and checks if the username exists, if it doesn’t, it returns a message to the receiveLogin Servlet that returns the message in an alert prompt that the username has already been taken (not the best security Standard) and returns the user to the index.jsp page to try again.

8c – If the username is in the database, the checkUsername stores that information in a temporary data with the username and password and calls the check Password method to see if the password matches what was provided.

8d - If they don’t match then the it returns a message to the receiveLogin Servlet that returns the message in an alert prompt that the password is incorrect (not the best security Standard) and returns the user to the index.jsp page to try again.

9a – Before the Servlet\_Snake\_Game page loads, it calls the ProcessScores class and gets the max high score and displays it for the user to see to know the top high score to beat.

11a – When the snake hits the wall or eats itself, the user has three options: Play Again, sign out, and check the high scores.

11b – If any of the options are clicked, the ProcessScores class is called which updates the database with the last score of the user and if there is a new high score.

13a – Before the Servlet\_High\_Score page is loaded, it connects with the database and gets the top 10 high scores with the associated username and displays it on a table for the user.

1. Post Conditions

* Each User can still access their accounts and play the game simultaneously for up to 12 users because of the Thread Connection Pool size.
* Each User must be signed in if they want their high score to be sent over to the database. This program doesn’t have full support for session authentication so directory traversal is possible.
* Users must play the game for their scores to be updated. If they don’t start playing and they go to another page then the database will not update any scores.

1. Supplemental Requirements

* There is no option for users who have forgotten their password to change it. This is something that can be added later as required.

1. Visual Model

See UML Sequence Diagram.

Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| V. | Date | Author | Description | Status |
| 1 | 3/22/2022 | Marvel Okafor | Finished up the use case | Completed |
|  |  |  |  |  |