Use Cases

Use Case 1: Log into the application **(Diagram 1)**
Author: Mikolaj Knap
Date: 27/11/2019
Purpose: Allow the user to securely log into their account using their password and name
Overview: The user upon startup of the application is directed to a login screen where their username and password is requested. Once the user tries to log in, the server system will then validate the password/username. If the authentication is successful, the user will be redirected to the start menu in the game. However, if the authentication is unsuccessful, the user will receive an error message stating that the provided credentials could not be authenticated. The user will be able to retry authentication into their account repeatedly until the user reaches a limit of tries to log in.
Cross-reference: requires authentication and signup
Actors: User, Server
Pre-conditions:

- The server must be up and ready to accept requests for validation
- The user must be connected to the server to be able to communicate with it

Post-conditions:

- The user upon successful authentication will be redirected to the start menu of the game
- If authentication is unsuccessful the user will stay on the login screen

Flow of events:

User Actions	Server Actions
1. The user opens up the game into the login screen	3. The server receives these credentials
2. The user enters their credentials to log in	4. The servers authenticates these credentials
6. The user receives the authentication message sent by the server	5. The server returns either a successful login message and redirects the player, or the server returns an error message

Alternative flow:
Step 4: The server cannot authenticate these credentials and an error message is displayed to the user
Step 4: The server has received these log in credentials too many times and has now locked out the player out of their account

Use Case 2: Have the user Sign up for the application **(Diagram 2)**
Author: Mikolaj Knap
Date: 27/11/2019
Purpose: Allow the user to create a new account
Overview: The user upon startup of the application is directed to a login screen where they can select the option to sign up for a new account. This will then redirect the user to a register screen where they will be able to sign up with a new username/password. Once the user signs up and the system confirms the credentials as valid, the new account will be added to the server's database. However, if the credential validation is unsuccessful the user will receive an error message stating that the provided credentials could not be validated. For validation the user's credentials will need to meet a certain set of requirements and the server check and protect against any potential malicious credentials.
Cross-reference: requires credential check to work
Actors: User, Server
Pre-conditions:

- The server must be up and ready to accept requests for validation
- The user must be connected to the server to be able to communicate with it

Post-conditions:

- Upon successful registration, the user will be redirected to the login screen
- If the registration is successful, the new account details will be stored on the server

Flow of events:

User Actions	Server Actions
1. The user opens up the client into the login screen	4. The server receives these credentials
2. The user goes to the sign up screen	5. The servers validates these credentials
3. The user enters their credentials to create a new account	6. The server returns either a successful registration message and redirects the player, or the server returns an error message
7. The user receives a message of a successful account creation, or an error message	

Alternative flow:
Step 4: The server cannot validate these credentials and an error message is displayed to the user

Use Case 3: Player changes direction of the snake**(Diagram 4)**
Author: Mikolaj Knap
Date: 27/11/2019
Purpose: Allow the player to change the direction of the snake by using their arrows keys to specify the new direction they want to go in
Overview: The player should be able to change the direction of the snake during a game by using their keyboard. During a game upon pressing the arrow button the snake should change direction and continue forward in that direction automatically. The player however will not be able to change the direction of the snake, to the opposite direction.
Cross-reference:
Actors: User
Pre-conditions:

- The player must be in an active game
- The player must be alive

Post-conditions:

- The snake will change direction based on the input by the user

Flow of events:

User Actions
1. User presses one of the arrow keys

Alternative Flow:

Step 1: The players tries to change direction, to the opposite direction the snake is heading in. This is impossible since the snake cannot go through itself and the game will simply ignore this key input

Step 1: The player changes direction twice rapidly, unless the snake collides with itself the game will treat this action as two seperate inputs and change direction twice

Use Case 4: Save Score**(Diagram 3)**
Author: Mikolaj Knap
Date: 27/11/2019
Purpose: Store the score the player has achieved once their game has ended
Overview: Once the player has finished their game, either through dying or actively ending the game, the score the player has achieved in that game will be sent to a server. The server will receive this score, store it, and update the user's score if necessary. If the score is 0 the server will not store it.
Actors: User, Server
Pre-conditions:

- The server must be up and ready to accept requests for validation
- The user must be connected to the server to be able to communicate with it
- .The user was in an active game that they just ended

Post-conditions:

- Upon successful registration, the user will be redirected to the log -in screen

Flow of events:

User Actions	Server Actions
1. The user ends their active game	3. The server receives the score
2. The user sends their score to the server to store	4. The servers updates the player's new highscore if necessary and records this score under the user's name
6. The user is put into the end game screen and shown their score	5. The server returns a successful message if saving the score went through

Alternative Flow:

Step 4: If the score received by the server is 0, the server will simply not store this as this is unnecessary since this will not be a new high score for a player ever.

Use Case 5: Player starts a new game **(Diagram 3)**
Author: Mikolaj Knap
Date: 27/11/2019
Purpose: Allow the user to startup a new game of snake
Overview: Once the player is logged into the game, they should have the option of starting a brand new game of snake. If the player decides they want to start a new game, they should be put into a fresh brand new game of snake.
Actors: User
Pre-conditions:

- The user must not be in an active game of snake
- The user must be logged into the server
- The user must be in the start menu of the game

Post-conditions:

- The user loaded into a new game of snake

Flow of events:

User Actions
1. User Presses the start button on the Start Menu
2. User is loaded into a new game of snake

Use Case 6: Player ends the game they are playing currently **(Diagram 3)**
Author: Mikolaj Knap
Date: 27/11/2019
Purpose: Allow the player to end the game they are currently in
Overview: During a game the player might be playing, the player will have the option of instantly stopping a game by pressing the specified end game button. Once this button is pressed the game will end and the player will be unable to resume. The user will be put into the end game screen where they will see their scores and have options as to whether they can restart the game or quit.
Cross-reference:
Actors: Player
Pre-conditions:

- The user must be in an active game to stop it
- The user's game must not be paused

Post-conditions:

- The user is put into the end game screen
- The user is displayed their final score for the game along with the top 5 high scores and the options to go to the main menu or exit the application

Flow of events:

User Actions	Server Actions
1. The user ends their active game	3. The server receives the score
2. The user sends their score to the server to store	4. The servers stores the player's score
5. The user is put into the end game menu and displayed their score along with the top 5 scores	

Diagrams

Overall Diagram

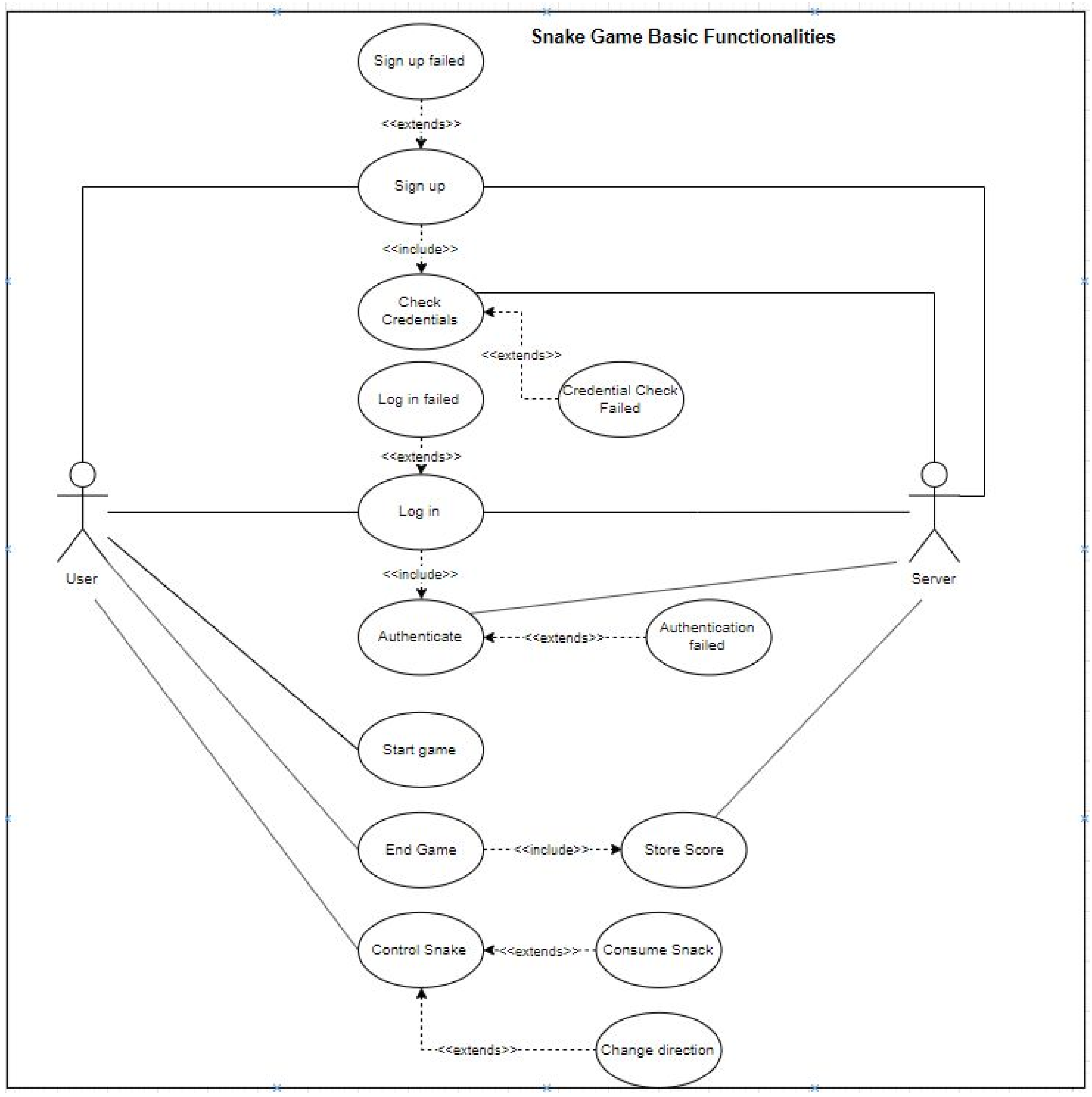


Diagram #1

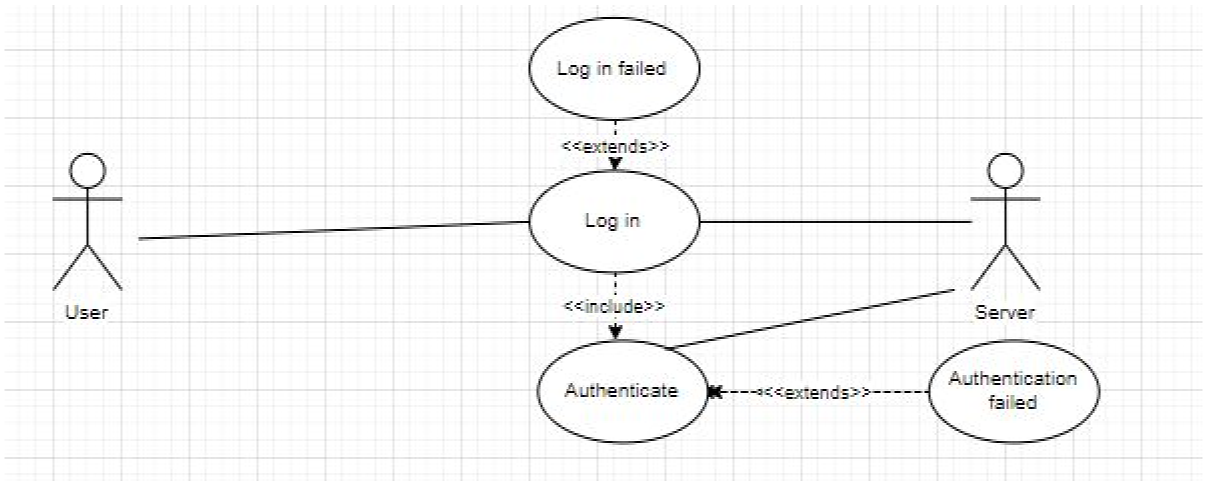


Diagram #2

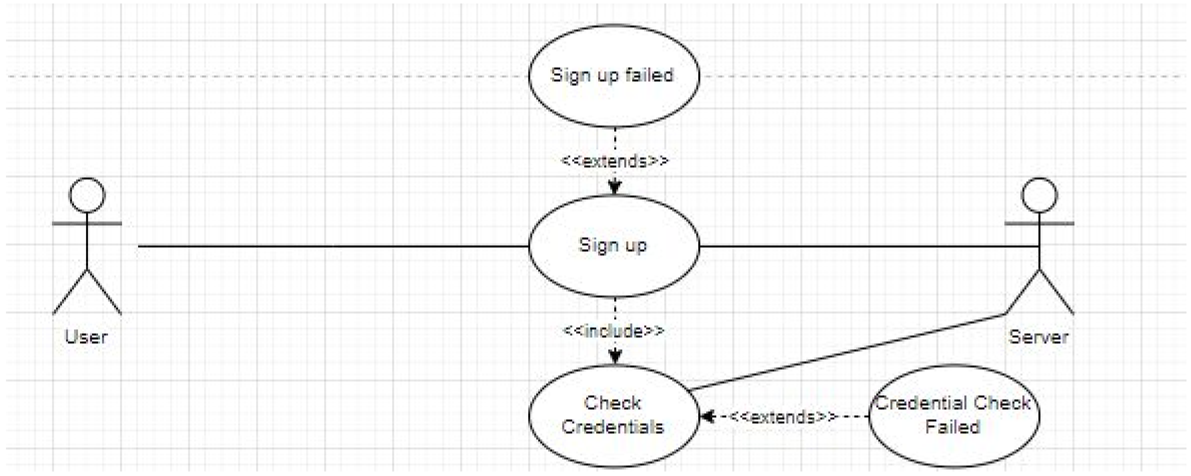


Diagram #3

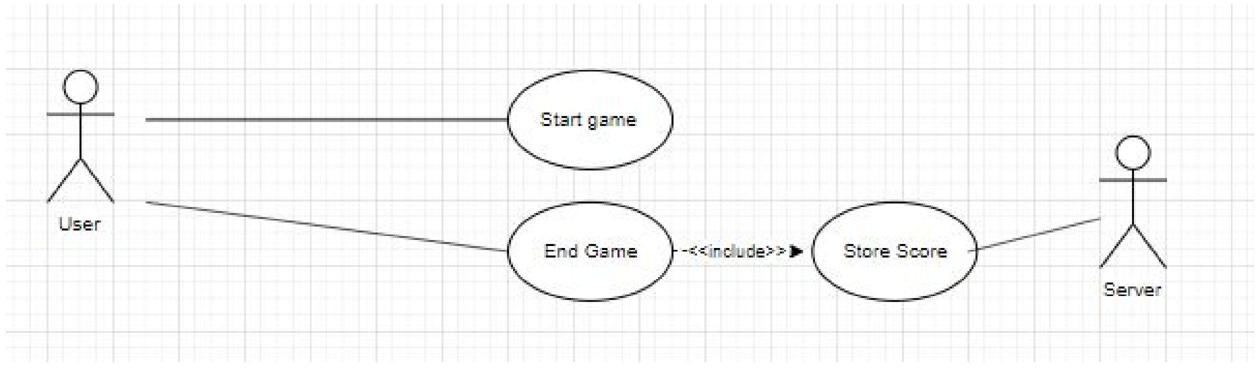
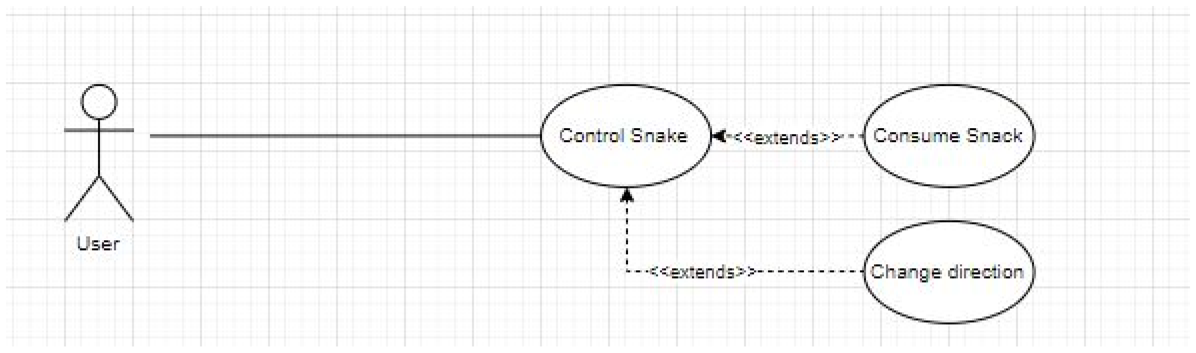


Diagram #4


User
Control Snake
<<extends>>
Consume Snack
<<extends>>
Change direction