Exercise 1

1. Functional requirements

We use the MoSCoW model for prioritizing requirements:

Must Have

- The game shall have a name;
- The player shall be able to launch the game;
- **The player shall be able to register a new account;**
- **The player shall be able to login using their account details;**
- **The player shall be able to start the game only if authentication is successful;**
- The game shall start with an empty game map;
- The game map shall be of at least 11 by 11 tiles;
- The game map shall cover a part of the window;
- The game map shall have clearly visible borders;
- The game shall start with a snake of starting size on the screen;
- The initial snake shall start moving when game starts;
- **The player shall be able to move the snake in a square pattern (left, right, up and down) using the keyboard;**
- The game shall not allow the snake to go outside the map borders;
- **The snake shall be able to eat food or other objects** (food must disappear when the snake touches it);
- The game shall place one food object on the arena;
- The game shall automatically generate food on the arena at a predefined frequency during the gameplay;
- The snake shall be able to grow;
- The snake shall grow by one cell when it eats one food object;
- The score shall increase by 10 when the snake eats one food object;
- **The player shall be able to stop a game of Snake that is currently in progress;**
- **The game shall end when a snake crosses itself;**
- **The game shall end when a snake hits the borders of the map;**
- The game shall set the player's score to 0 when starting the game;
- The game shall update the user score throughout gameplay;
- The game shall show the user score throughout gameplay;
- The game shall save the score of each game;
- The player shall be able to write a nickname that would be associated with their score at the end of each game ;
- The player shall be able to see the top 5 leaderboard of the scores of all players at the end of each game;
- The player shall be able to see their personal highscore (only one highest score);

The requirements **in bold** from this list are the ones we chose for drawing Use Case Diagrams.

Should Have

- The game shall have a theme;
- The game shall show a loading screen for some defined short period of time when the application is launched;
- The game shall show a login screen when the game is opened (or after the loading screen);
- The player shall be provided with instructions of the game;
- The player shall be able to choose from different rectangular game maps;
- The player shall be able to restart a game;
- The player shall be able to pause a game that is currently in progress;
- The player shall be able to resume a previously started game;
- The player shall have access to a personal leaderboard;
- The player shall be able to make their username be the nickname associated with their score by ticking a checkbox;
- The player shall be able to change the keys with which the snake is controlled (ex: change from arrow keys to WASD keys);
- The game shall have 3 or more different snake skins that the player can choose from;
- The snake shall be able to come out from the opposite side of the arena if there is no border;
- The game shall allow the snake to change movement direction by 90 degrees with each (by default it allows moving in opposite directions does);
- The game shall increase the speed of the snake gradually during gameplay;
- The game shall contain the following special interactive elements:
    - a power-up that slows down the snake;
    - a downgrade that speeds up the snake;
    - obstacles that end the game if the snake hits it;
- The game shall automatically generate all interactive elements (food, downgrades, power-ups, obstacles, etc.) on the arena during the gameplay at a predefined frequency;
- The game shall automatically increase the frequency of generating interactive elements as the game progresses;

Could Have

- The game shall have modes of different difficulties (for example: hard, normal and easy);
- The game shall have modes for different snake food preferences;
- The game shall have modes for different environments;
- The game shall have a draw mode (where the snake’s head acts like a paintbrush);
- The game shall change the colors of some elements for different modes of the game;
- The game shall generate different interactive elements for different modes of the game;

- The game shall contain special interactive elements that:
    - teleport the snake to a different location in the arena;
    - change the controls of the snake (ex: make the snake go left when the user hits the right arrow key);
    - change the colors of other elements in the game;
    - change the arena of the game;
    - trim the length of the snake;
    - change the moving pattern of the snake;
    - do not terminate the game once when it normally should (like a second life);
- The game shall display the interactive elements are affecting the snake during gameplay;
- The interactive elements shall have textures (shall not be just solid color blocks);
- The game shall have background music;
- The player shall be able to turn the music and sounds of the game on or off;
- The game shall play different sounds for different events:
    - Sound for start of the game
    - Sound for eating food
    - Sound for powerups (or just positive events)
    - Sound for downgrades (or just negative events)
    - Sound for game over
- The game shall have a mode that allows two players to play on the same device;
- The game shall have additional rules that allow two players to compete with each other in one game (example: a rule that would allow snakes to eat other snakes);
- The player shall be able to choose more complex game maps of different shapes (not rectangular) to play on;
- The game map shall change in size during gameplay;
- The game map shall change in shape during gameplay;
- The game shall have an online multiplayer mode;
- The game shall allow the user to create their own snake skin;
- The game shall allow the user to save their custom snake skin;
- The game shall allow the user to use snake skins created by other users;

Won't Have

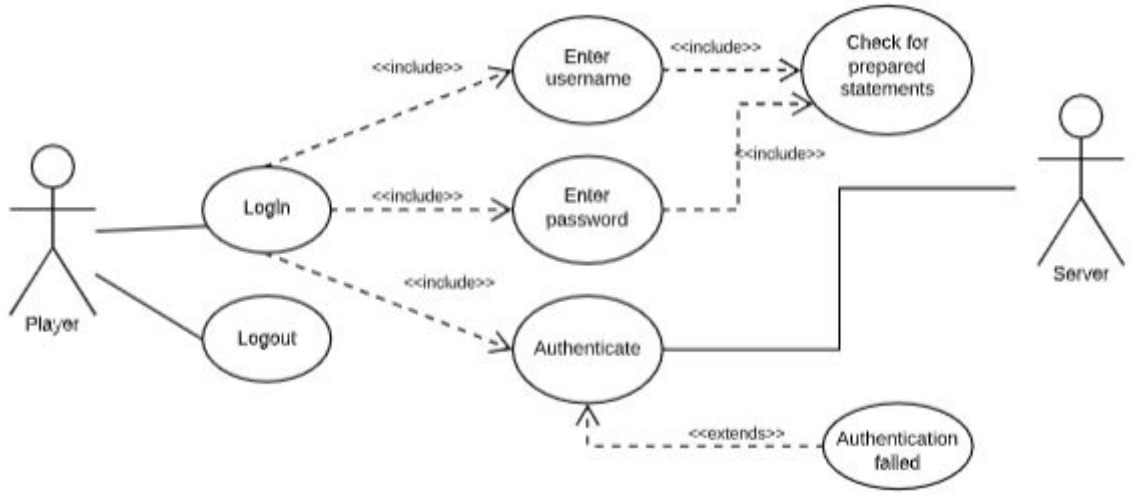
- The player shall be able to change the background of the game window by choosing an image from their computer files;
- The player shall be able to change the background of the game map by choosing an image from their computer files;
- The player shall be able to upload snake skins by choosing an image from their computer files;
- The player shall be able to model their own game map;
- The player shall be able to use the mouse to control the snake;
- The player shall be able to customize the effects of default interactive objects (powerups, downgrades, food etc.);

2. Non-functional requirements

- The game shall be implemented using the libgdx game development framework.
- The game shall be playable on Windows (8 or higher) and Mac OS X (10.8 and higher).
- The game shall be implemented in Java.
- The game shall use SQL and JDBC driver.
- The game shall be playable on computers and laptops. Phones and other devices will not be supported.
- The game shall have a contrasting color scheme for accessibility reasons.
- Our team shall apply SCRUM methodology while working on this project.
- The architecture of the system shall be very clear and well motivated.
- All features are implemented with approval from Sara, our client.
- A first fully working version of the game shall be delivered on the 6th of December 2019.
- The final version of the game shall be delivered on the 24th of January 2020.
- The final version of the game shall be presented on 21st of January 2020 with all team members present.
- The game system shall use SQL and JDBC drivers.
- The input of the game shall use prepared statements in Java to avoid code-injection vulnerabilities.
- Each delivered iteration of our game must have at least 75% meaningful line test coverage. The tests must make sure that the methods are not only executed but that they behave as expected. For this JUnit test framework and the code coverage utility JaCoCo shall be used.
- The source code shall be versioned using the TU Delft distribution of GitLab.
- Sequence diagrams and high-level documents shall be used to design integration/system level tests. Mutation testing tools shall be used to improve assertions.
- The project shall use continuous integration and include test and static tools output in the building process. For this, the code shall be analysed with the support of three static analysis tools, namely CheckStyle, FindBugs and PMD.

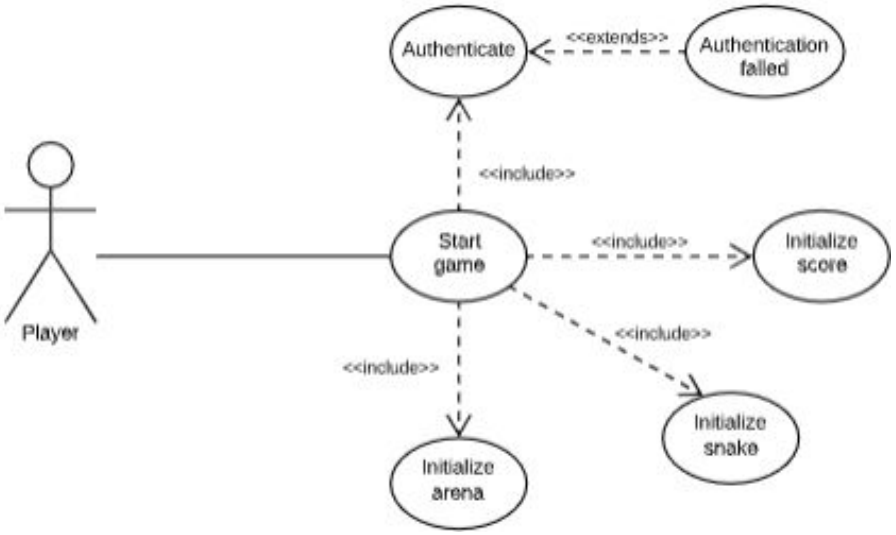
Exercise 2

1. User login



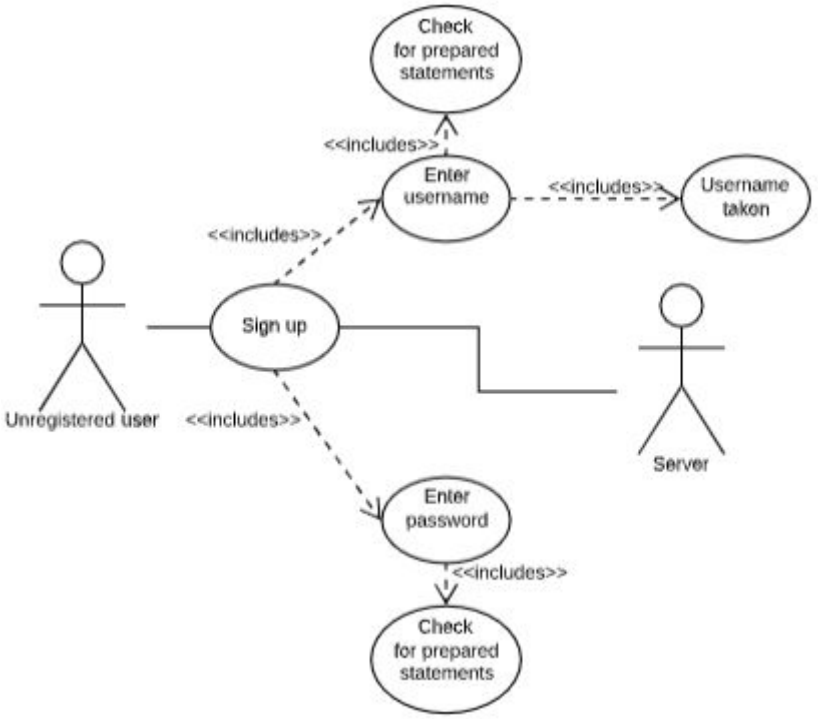
Author/date	Gabriele, Mihai, Mirijam, Sanjay, Roman, 21 nov 2019
Modification/Date	24 nov 2019
Purpose	Player log in and log out
Overview	The player tries to log in by entering their username and password. Then the server checks if the user exists and the password is correct. When authentication is successful the user is logged in and able to play the game.
Cross references	The player shall be able to login using their account details.
Actors	Player who has not logged in yet Server (local)
Pre-conditions	Registered user has launched the game and not logged in yet.
Post-conditions	User is logged in.
Normal flow of events	● User enters username and password, ● Server authenticates given username and password ● User is logged in.
Alternative flow of events	● User enters username. ● User enters password. ● Authentication fails.
Exceptional flow of events	Server is down and no player can log in and presents message: ”Server is down”

2. Starting game



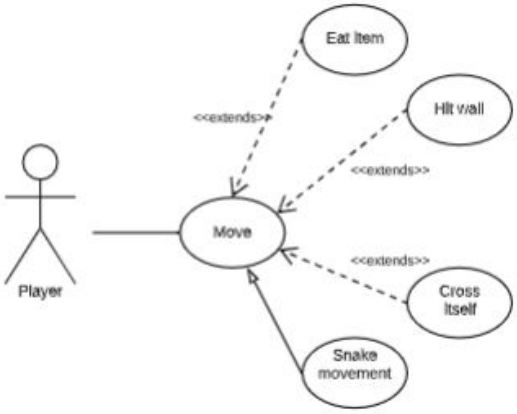
Author/date	Gabriele, Mihai, Mirijam, Sanjay, Roman
Modification/Date	24 nov 2019
Purpose	Starting the game
Overview	After the user authenticates him/herself, the score and snake gets initialized.
Cross references	The player shall be able to start the game only if authentication is successful.
Actors	Registered player
Pre-conditions	User has launched the game and logged in
Post-conditions	After authenticating, the score and snake get initialized.
Normal flow of events	● User gets authenticated. ● Score, map and snake gets initialized. ● Game starts.
Alternative flow of events	Authentication fails and thus, the game doesn't start.
Exceptional flow of events	Game crashes and unexpected behavior occurs (ex: its elements are not initialized)

3. User sign up



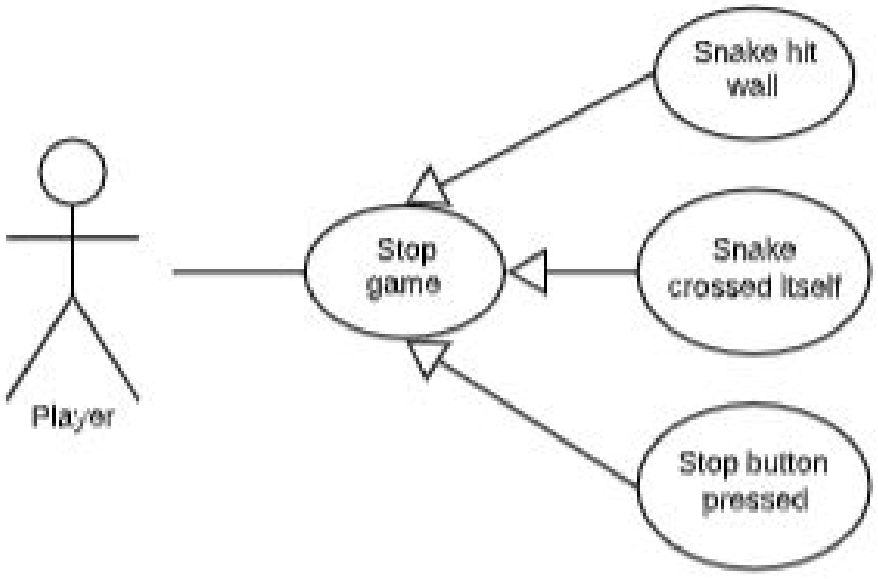
Author/date	Gabriele, Mihai, Mirijam, Sanjay, Roman
Modification/Date	24 nov 2019
Purpose	Sign up a new user
Overview	User can sign up by choosing a username, which has not been used yet, and password. The username and password are then passed to the server.
Cross references	The player shall be able to register a new account.
Actors	Unregistered user Server (local)
Pre-conditions	User has not been registered yet.
Post-conditions	User has an account to which they can log in.
Normal flow of events	● Unregistered user enters a username ● Unregistered user enters a password ● Server creates an account
Alternative flow of events	● Unregistered user enters an existing username ● Server declines to create a new account
Exceptional flow of events	Server is down and no new account can be created. The error message “Server is down” is displayed.

4. Movement



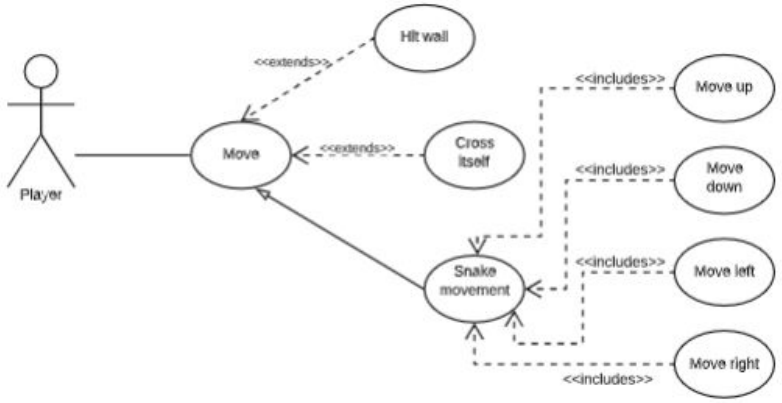
Author/date	Gabriele, Mihai, Mirijam, Sanjay, Roman
Modification/Date	24 nov 2019
Purpose	Interact with map elements when moving snake
Overview	The player can move the snake around the map and this leads different snake actions.
Cross references	The game shall end when a snake crosses itself. The game shall end when a snake hits the borders of the map. The snake shall be able to eat food or other objects
Actors	Player
Pre-conditions	Player has to be logged in and game has to be started
Post-conditions	Player moves the snake on the map which lets it interact with other elements like the wall and itself.
Normal flow of events	● The player is controlling the snake and can make it either cross itself, eat an item or hit a wall
Alternative flow of events	● Player presses random keys which are not w, a, s, d or up, down, left, right arrows and, therefore, nothing happens.
Exceptional flow of events	Game crashes. Unexpected behavior.

5. Stop game



Author/date	Gabrielė, Mihai, Mirijam, Sanjay, Roman
Modification/Date	28 nov 2019
Purpose	Stopping the game
Overview	The game can be stopped due to the player losing the game or the player intentionally stopping it.
Cross references	The player shall be able to stop a game of Snake that is currently in progress.
Actors	Player
Pre-conditions	The game has been started and is yet in progress. User is registered.
Post-conditions	The game processes stop (e.g., movements, interactions), but the UI allows to restart the game.
Normal flow of events	● Player is playing the game; ● Player invokes the stop condition (e.g., hits the wall or stop button); ● The game processes stop; ● UI displays the option to enter the leaderboard and restart the game.
Alternative flow of events	● Player is playing the game; ● Player just closes the game window;
Exceptional flow of events	Game crashes. Unexpected behavior.

6. Move snake



Author/date	Gabrielė, Mihai, Mirijam, Sanjay, Roman, 21 nov 2019
Modification/Date	24 nov 2019
Purpose	Moving the snake.
Overview	User can move the snake using up, down, left and right arrow keys.
Cross references	The player shall be able to move the snake in a square pattern (left, right, up and down) using the keyboard.
Actors	Player
Pre-conditions	Game has to be started and ongoing. There is a moving snake on the map.
Post-conditions	The snake gets moved in the direction according to the arrow key the player pressed.
Normal flow of events	● The player presses an arrow key (up, down, left, right). ● Snake's movement changes direction accordingly. ● The snake advances in that direction.
Alternative flow of events	● Snake is on the map. ● A button (up, down, left, right) gets pressed. ● Snake moves accordingly but hits a wall or crosses itself.
Exceptional flow of events	Game crashes. Unexpected behavior.

Exercise 3

The requirement we focused on implementing for this sprint:

- ➔ The player shall be able to launch the game;
- ➔ The game shall start with an empty game map;
- ➔ The game map shall be of at least 11 by 11 tiles;
- ➔ The game shall start with a snake of starting size on the screen;
- ➔ The initial snake shall start moving when game starts;
- ➔ The game shall not allow the snake to go outside the map borders;

- ➔ The player shall be able to move the snake in a square pattern (left, right, up and down) using the keyboard;
- ➔ The snake shall be able to grow;