Criterion B: Design Stage

Within the design overview, I will be explaining the following sections:

1. **Overall structure (User perspective)**
   a) General overview of the user interface
   b) Individual components of the user interface
   c) Flow of user interaction
2. **Internal structure (Developer perspective)**
   a) Libraries Used
   b) Program files
   c) Program functions
   d) Program classes
   e) Detailed flowchart of program
   f) Developer test plan

Overall structure (User perspective)

a) General overview of the user interface

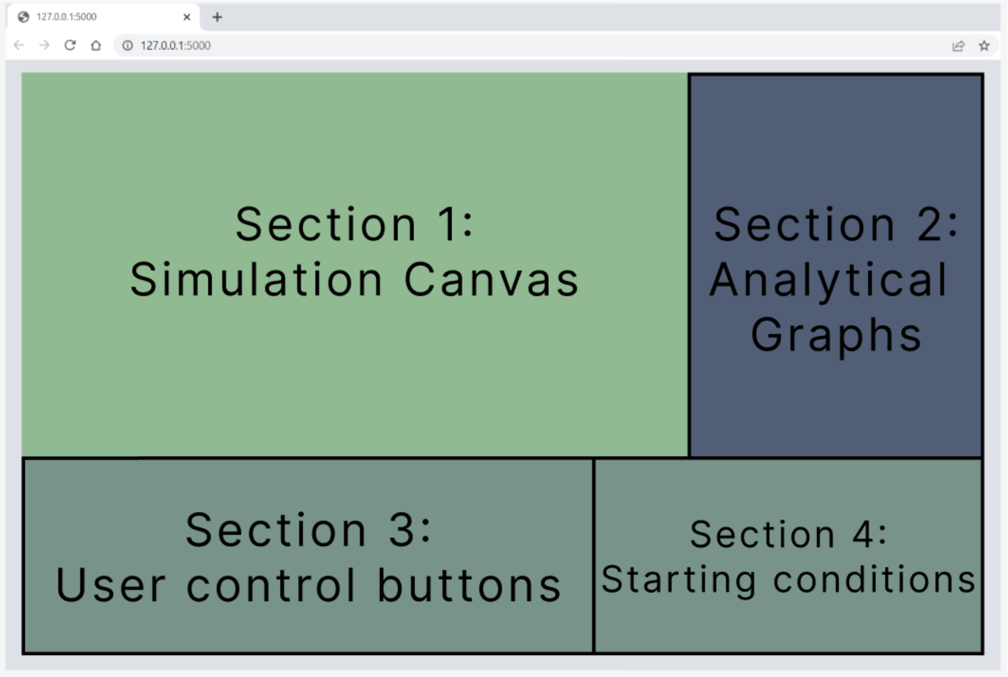
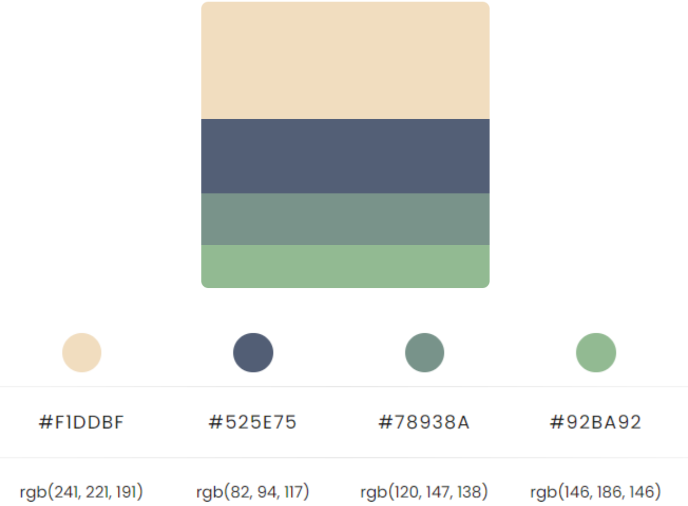


Fig 1.0 - User interface

The product will be built on a fullscreen webpage, put together using Bootstrap.html. It will contain 4 main sections as shown in fig 1.0



●	●	●	●
#F1DDBF	#525E75	#78938A	#92BA92
rgb(241, 221, 191)	rgb(82, 94, 117)	rgb(120, 147, 138)	rgb(146, 186, 146)

Fig 1.1 - Color palette of the webpage

b) Individual components of the user interface

Section 1: Simulation Canvas

A 2D visualisation of the simulated environment. The entire population of rabbits and grass will interact within this canvas. Elements within the canvas will be drawn using p5.js

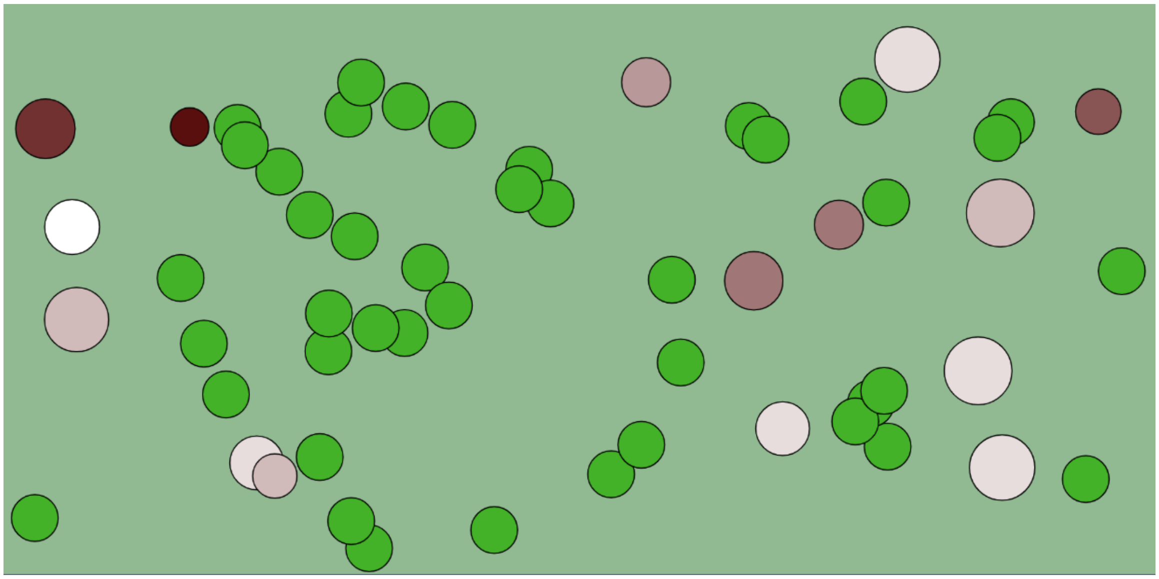


Fig 1.2 - Example of what the canvas will look like
(circles are placeholders for the elements of the simulation)

Section 2: Analytical Graphs

Graphs that show data about the population over time. They will update automatically at consistent time intervals.

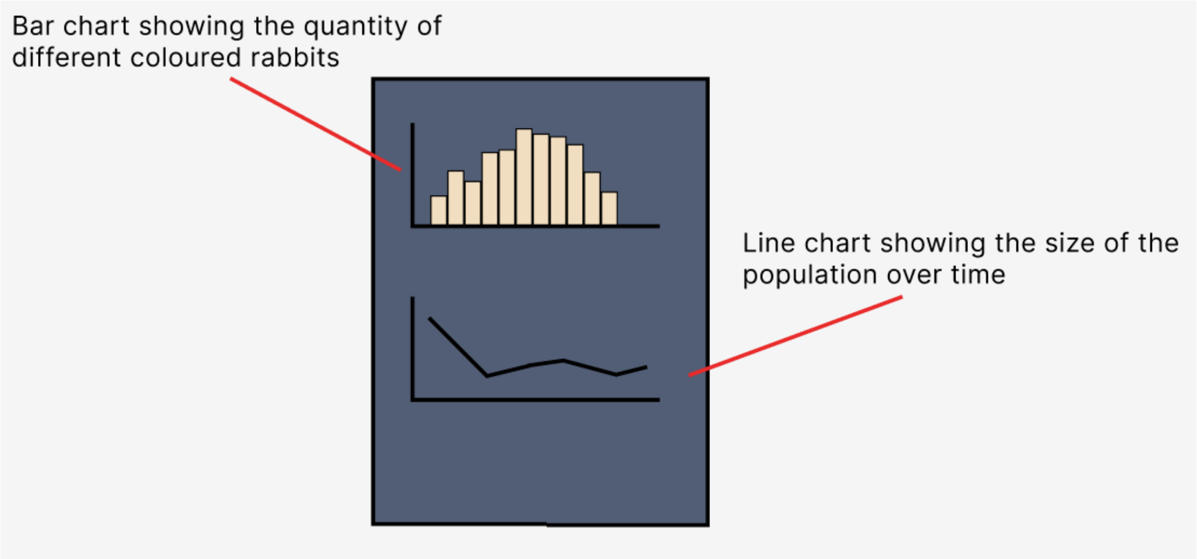

Bar chart showing the quantity of different coloured rabbits
Line chart showing the size of the population over time

Fig 1.3

Section 3: User control buttons

Various buttons for the user to control the simulation. The function and name of each button is shown below.

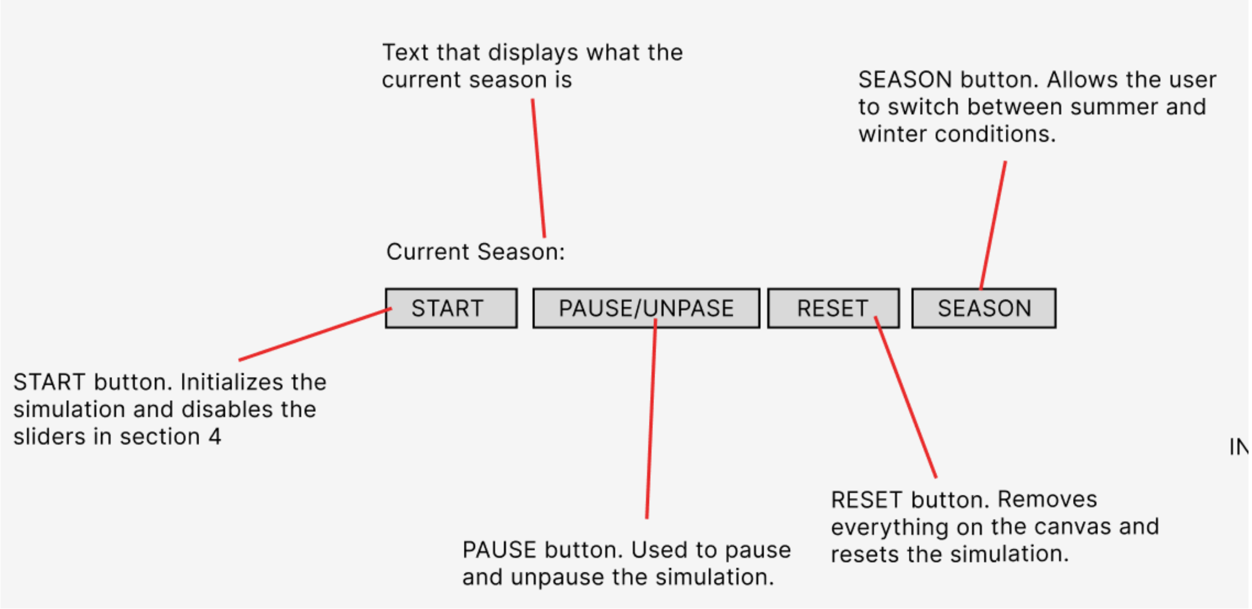

Text that displays what the current season is
SEASON button. Allows the user to switch between summer and winter conditions.
Current Season:
START
PAUSE/UNPASE
RESET
SEASON
START button. Initializes the simulation and disables the sliders in section 4
IN
RESET button. Removes everything on the canvas and resets the simulation.
PAUSE button. Used to pause and unpause the simulation.

Fig 1.4

Section 4: Starting conditions

Slide bars for the user to input starting values. These slide bars will be disabled once the START button is pressed, and re-enabled if the RESET button is pressed.

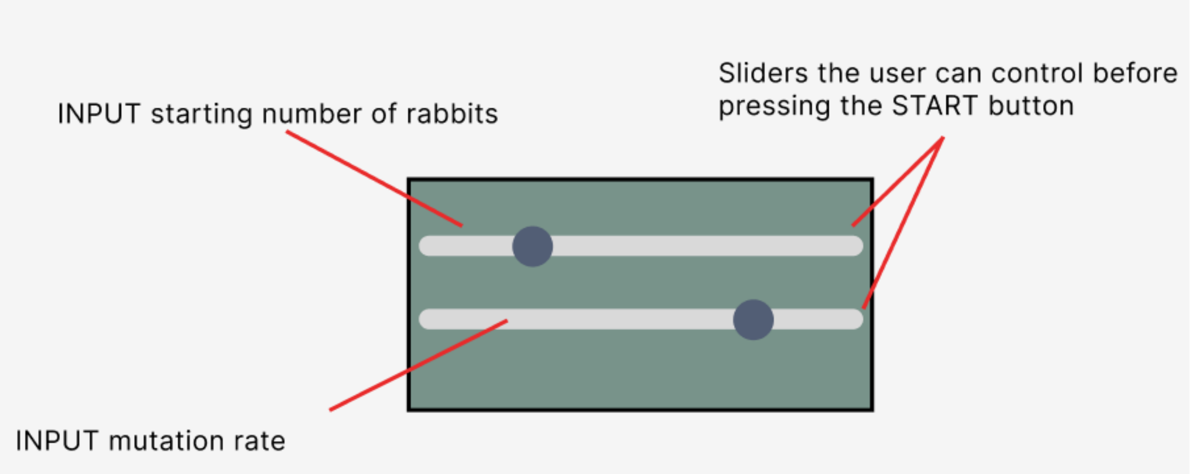

INPUT starting number of rabbits
Sliders the user can control before pressing the START button
INPUT mutation rate

Fig 1.5

c) Flow of user interaction

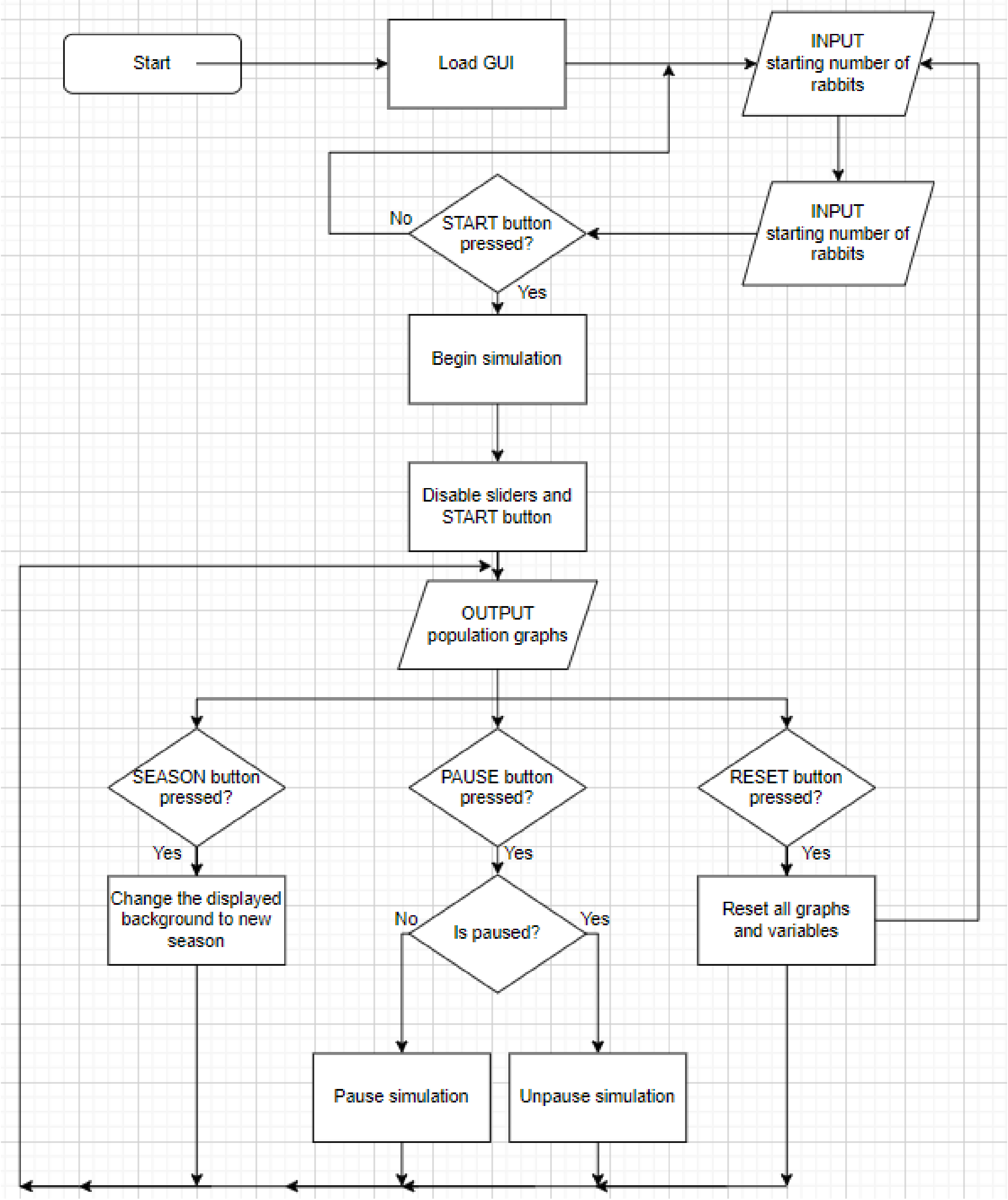


Fig 1.6 - Overall program flow

1. Internal structure (Developer perspective)

a) Libraries used

Library	Purpose
Flask	Creates virtual environment which is used to serve the application.
p5.js	Used to draw the rabbits, hawks, and grass on the canvas.
Chart.js	For creating dynamic graphs based on the current state of the population.
Bootstrap	Provides the website framework and allows for customisation.

b) Program files

Front end

File name	Purpose
sketch.js	Constantly draws and updates the canvas while the simulation is running.
home.html	Determines the structure, layout, and buttons on the generated user interface.
base.html	Links the various files within the folder together and grants access to the various libraries used.
style.css	Formats the webpage with specific fonts, colours, margins etc.
right_sliders.html	Sliders on bottom right of the web page for the user to input some starting values for the simulation.
right_graphs.html	Graphs on top right of the web page that display data about the population.

Back end

File name	Purpose
program.js	Where the main program and genetic algorithms will be carried out. Contains the rabbit, grass and hawk classes, as well as functions for each button on the GUI.
__init__.py	Initialises the virtual environment with flask.
routes.py	Provides a route for the root URL of the web application.

c) Program functions

Within the file “sketch.js”:

Function name	Purpose
setup()	Creates the canvas on which the simulation will be running.
draw()	Draws the elements of the simulation on the canvas and updates each of them if necessary.

Within the file “program.js”:

Function name	Purpose
start()	Begins the simulation based on the starting values input by the user and disables the input sliders.
LoadGraphics()	From the “graphics” folder, load all of the sprites for the rabbits, hawks, and grass into lists
addRabbits(n)	Spawns “n” number of rabbits into the simulation at random locations on the canvas.
addGrass(n)	Spawns “n” number of grass into the simulation at random locations on the canvas.
RandomBinary()	Generates a random 21 bit binary that represents the rabbit’s

	DNA.
halt()	Pauses or unpauses the simulation.
clearance()	Resets the simulation, deletes all the current elements, clears the graphs, re-enables the input sliders and start button.

d) Program classes

Within the file “program.js”:

Class: Rabbit		
Methods	**Name**	**Description**
	chooseaction()	Decides an action for the rabbit to take between: *Eat, Reproduce, Move, Find Direction.*
	update()	Carries out any action the rabbit takes, and updates its variables accordingly.
	choosedirection()	Determines the directional vectors for the rabbit’s next movement.
	mutate()	Takes the DNA and randomly flips a given number of bits depending on the variable “mutation_rate” (input by the user).
	draw()	Draws the sprite of the rabbit rabbit at a given size and location on the canvas depending on its colour
Variables	**Name**	**Description**
	dna	21 bit binary number that determines the colour, diameter, energy gain, intelligence, and speed of the rabbit. Passed on during reproduction.
	colour	Determines the colour of the rabbit ranging from white (255,255,255) to dark brown (90,15,15).
	diameter	Determines the size of the rabbit. Values range from

	32 - 63 pixels.
energygain	Increases the amount of energy a rabbit gains from a piece of food by a certain amount. Values range from 0 - 31.
intelligence	Determines the rabbit’s ability to pathfind towards food and potential mates. Values range from 0 - 15.
speed	Determines how far a rabbit is able to move in a given time frame. Value range: 0 - 15.
energy	Energy level of the rabbit. The rabbit will die if this drops below 0.
maxsize	The maximum size that the rabbit can grow to.
x	X coordinate of the rabbit.
y	Y coordinate of the rabbit.
vx	Horizontal velocity vector.
vy	Vertical velocity vector.
offset	Offsets the timing of the rabbit’s movement by a random amount to emulate the sporadic movement of real rabbits.
alive	A boolean value that tells you whether the rabbit is alive or not.
targeted	A boolean value that tells you whether the rabbit is currently being targeted by a hawk.
img	a list containing the sprites for the rabbit depending on its colour

Class: Hawk

Methods	**Name**	**Description**
	update()	Continuously updates the hawk’s position so that it moves towards its target. Also carries out the act of eating the rabbit.
	draw()	Draws the hawk at a given size and location on the canvas.
Variables	**Name**	**Description**
	target	The rabbit that this hawk is targeting
	x	X coordinate of the hawk.
	y	Y coordinate of the hawk.
	speed	The speed at which the hawk moves at.
	target_colour	The colour of the hawk’s target

Class: Grass		
Methods	**Name**	**Description**
	draw()	Draws the correct sprite for grass at a given size and location on the canvas depending on the current season
Variables	**Name**	**Description**
	x	X coordinate of the grass
	y	Y coordinate of the grass
	img	a list containing the sprites for the grass

e) Detailed flowchart of program

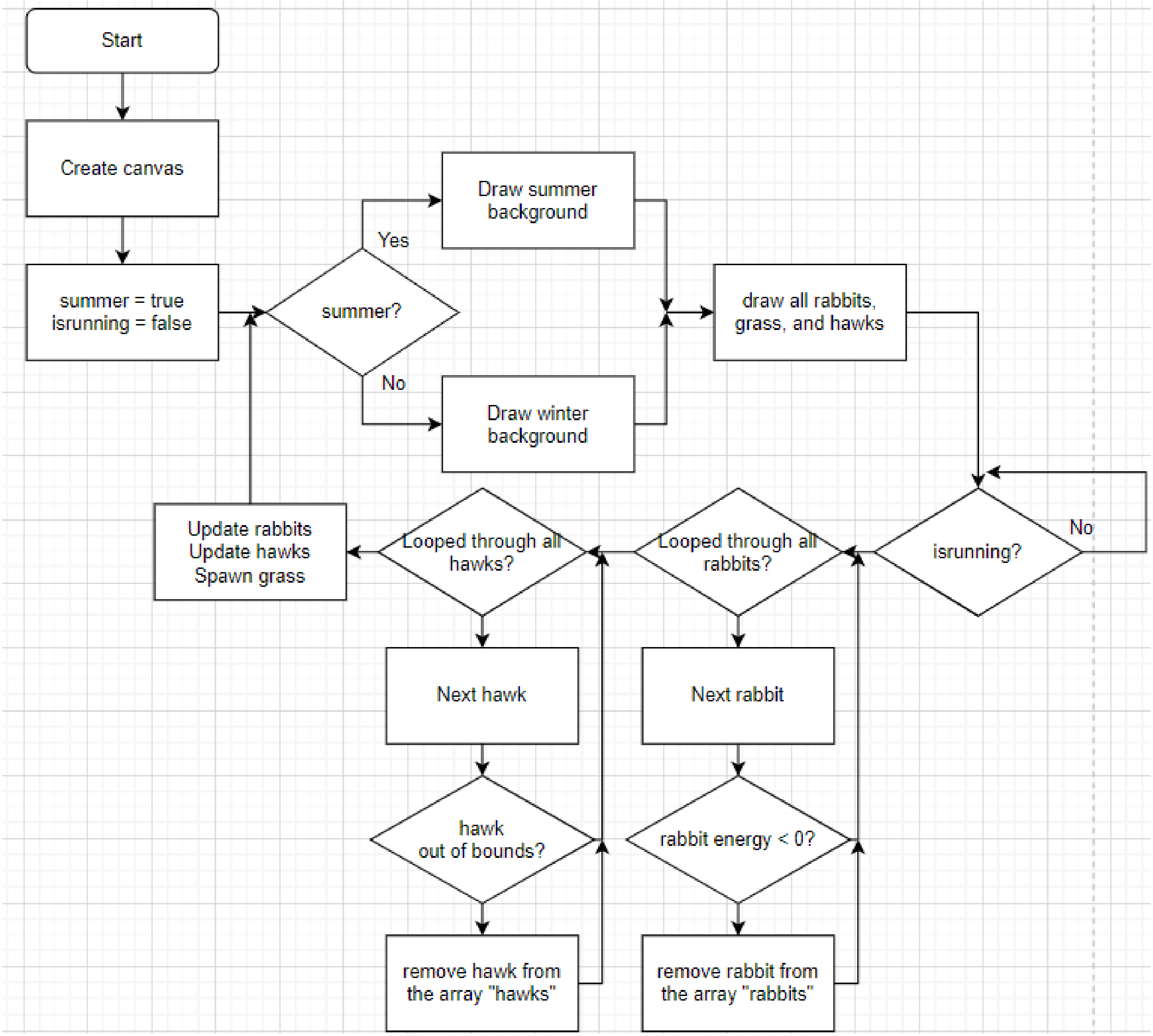
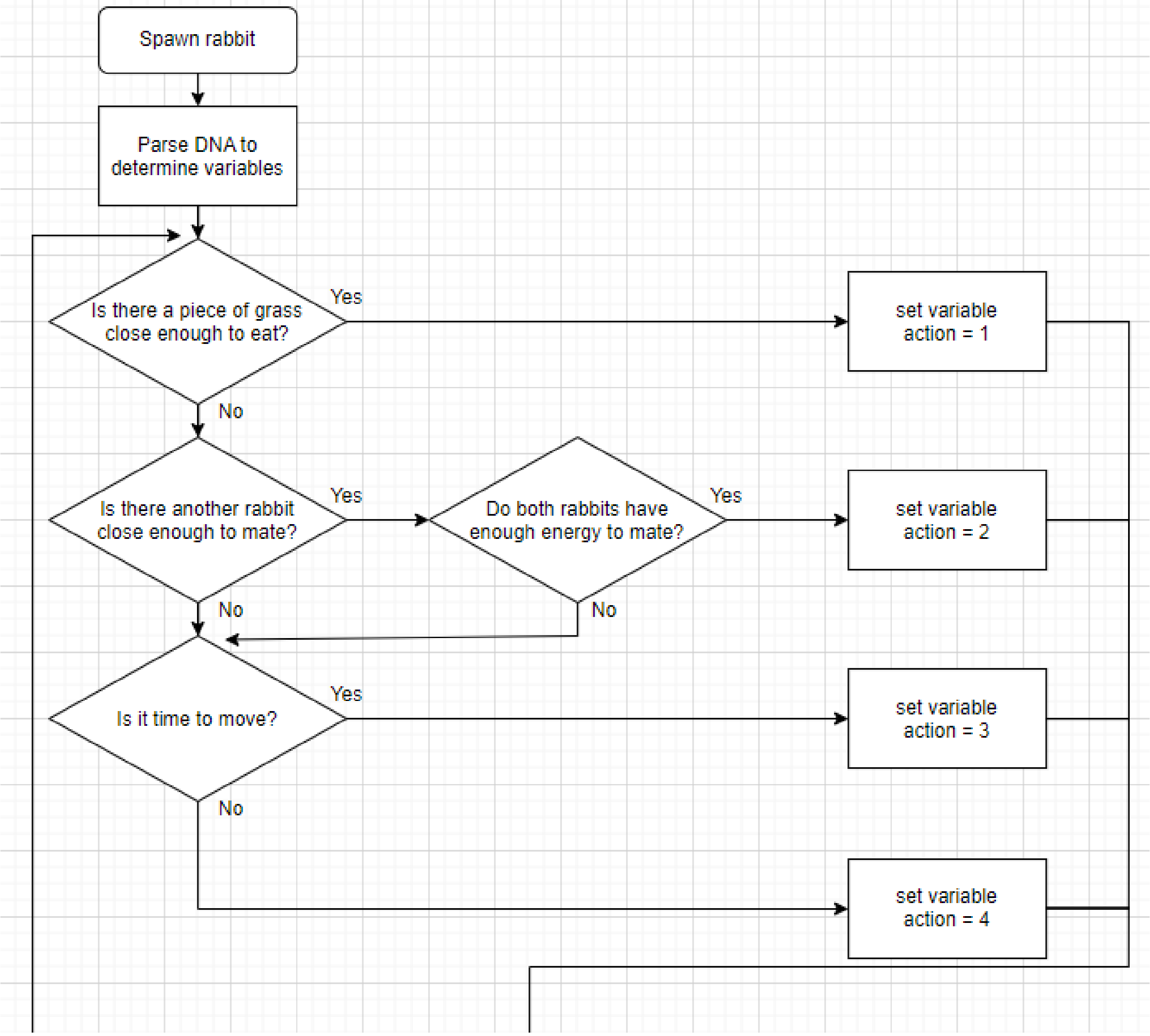


Fig 2.1 - Overall flow of the simulation


Spawn rabbit
Parse DNA to determine variables
Is there a piece of grass close enough to eat?
Yes
set variable action = 1
No
Is there another rabbit close enough to mate?
Yes
Do both rabbits have enough energy to mate?
Yes
set variable action = 2
No
No
Is it time to move?
Yes
set variable action = 3
No
set variable action = 4

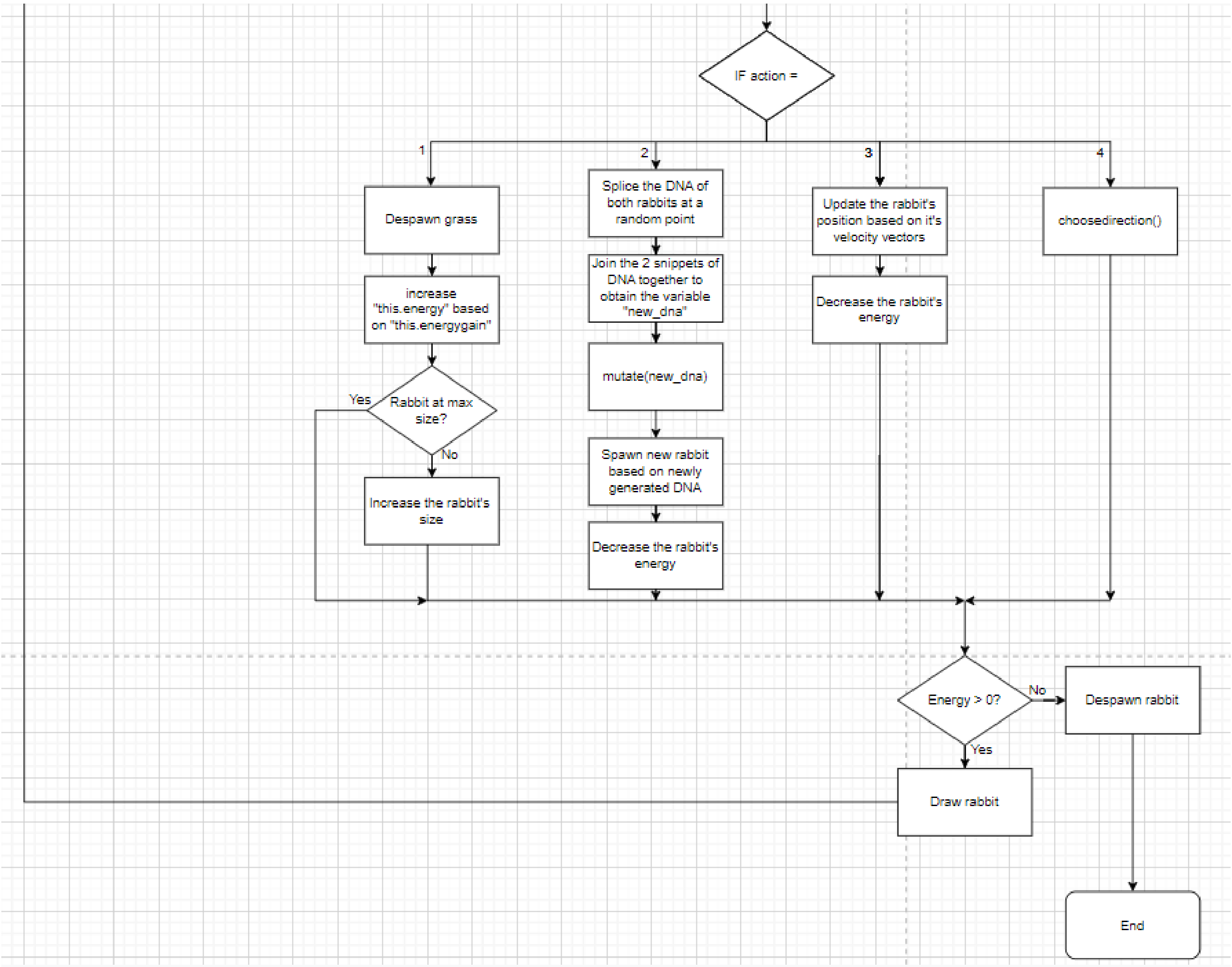

IF action =
1
2
3
4
Despawn grass
increase "this.energy" based on "this.energygain"
Rabbit at max size?
Yes
No
Increase the rabbit's size
Splice the DNA of both rabbits at a random point
Join the 2 snippets of DNA together to obtain the variable "new_dna"
mutate(new_dna)
Spawn new rabbit based on newly generated DNA
Decrease the rabbit's energy
Update the rabbit's position based on it's velocity vectors
Decrease the rabbit's energy
choosedirection()
Energy > 0?
No
Despawn rabbit
Yes
Draw rabbit
End

Fig 2.2 - Flowchart of the Rabbit class (split into 2 images)

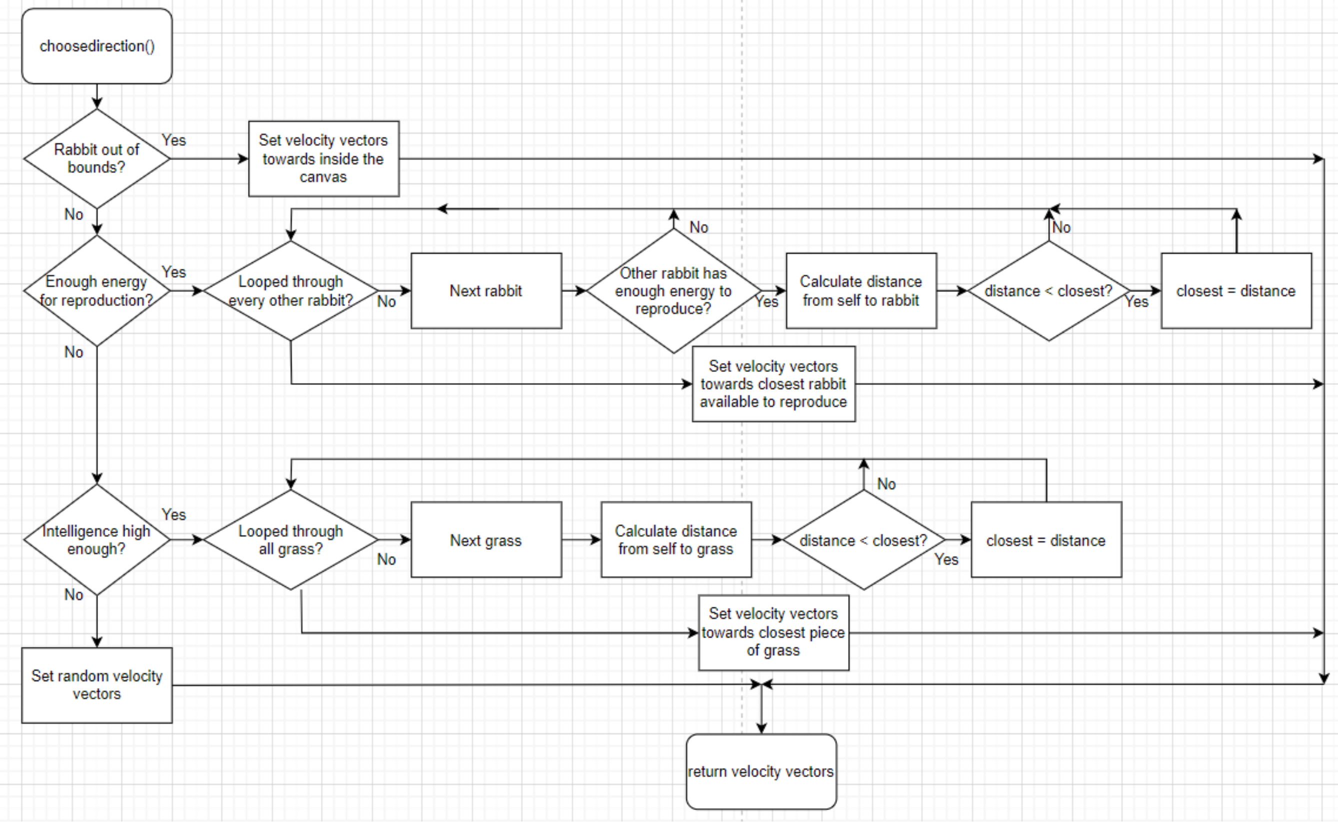

choosedirection()
Rabbit out of bounds?
Yes
Set velocity vectors towards inside the canvas
No
Enough energy for reproduction?
Yes
Looped through every other rabbit?
No
Next rabbit
Other rabbit has enough energy to reproduce?
No
Yes
Calculate distance from self to rabbit
distance < closest?
No
Yes
closest = distance
Set velocity vectors towards closest rabbit available to reproduce
No
Intelligence high enough?
Yes
Looped through all grass?
No
Next grass
Calculate distance from self to grass
distance < closest?
No
Yes
closest = distance
Set velocity vectors towards closest piece of grass
No
Set random velocity vectors
return velocity vectors

Fig 2.3 - Flowchart of the choosedirection() function under the Rabbit class

f) Developer test plan

Test Number	Aspect tested	Test plan	Expected result
1	Generated user interface loads properly	Open the webpage	The page is layout exactly as shown in the **Overall structure (User perspective)** section.
2	START button	Start the simulation	Rabbits and grass are spawned on the canvas and the simulation begins. The sliders should become disabled.
3	“Starting number of rabbits” slider	Attempt to use the slider and start the simulation	The slider should function properly and display the current value. Once the simulation is started, it should spawn the correct number of rabbits.

4	“Mutation rate” slider	Attempt to use the slider and start the simulation	The slider should function properly and display the current value. Mutations in DNA should occur based on the value input by this slider.
5	SEASON button	Press the SEASON button multiple times	The background should change colour depending on the season. It should change back and forth between two colours.
6	Season display text	Press the SEASON button multiple times	The display text should correctly tell us what the current season is. It should change every time the SEASON button is pressed.
7	PAUSE button	Press the PAUSE button twice	The simulation should get paused and then unpaused.
8	RESET button	Press the RESET button	All the elements within the canvas disappear, and the sliders become enabled again.
9	Colour bar graph	Observe the graph while the simulation is running	This graph should display the number of rabbits alive for each colour, updating constantly.
10	Population line graph	Observe the graph while the simulation is running	This graph should display the correct number of rabbits alive over time and update at consistent time intervals.
11	Rabbit movement	Observe the rabbits during the simulation	Each rabbit is able to move around freely at set intervals. They should not move out of bounds of the canvas.
12	Rabbit eating	Observe the rabbits during the simulation	When a rabbit comes into contact with a piece of grass, the grass should disappear.
13	Rabbit reproduction	Observe the rabbits during the simulation	Rabbits are able to create offspring with other rabbits.