User Requirements Specifications

Hyena Crossing

08-March-20

Version 1.3

Introduction

This document is going to cover all the required features that the user has specified from our application.

We are going to represent those requirements in a form of use cases.

Table of Contents

Non-Functional Requirements

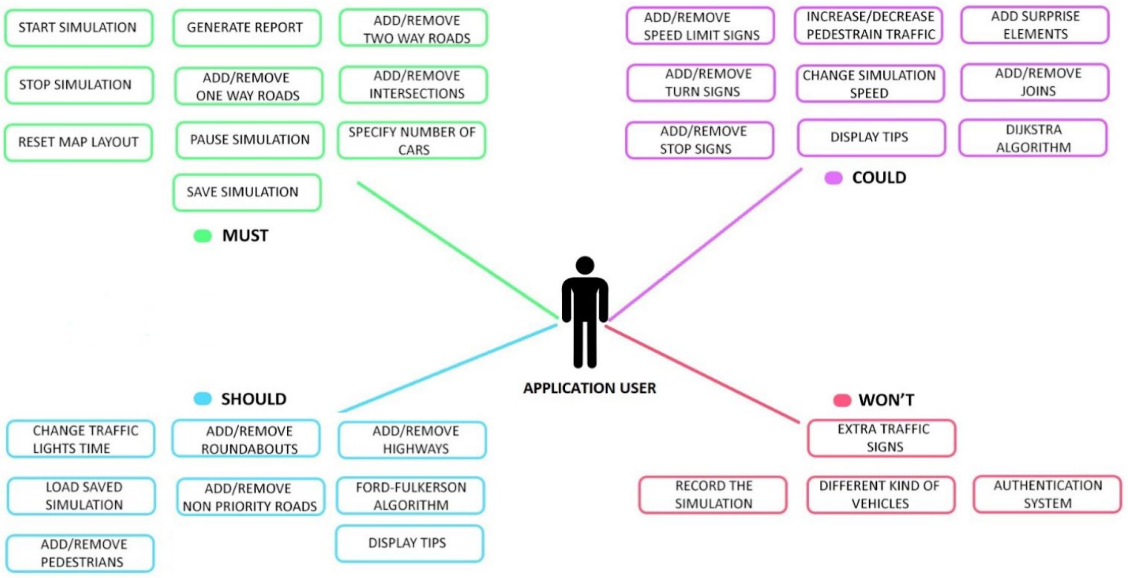
- **Robustness** - the application has to work without errors when executed.
- **Efficiency** - the application has to work smoothly with small response time
- **Usability** - the application needs to have user-friendly ui
- **Readability** - the code of the application has to be structured and the solutions justified
- **Testability** - the application has to be tested with test cases
- **Accessibility** - the application has to work on Windows

Functional Requirements

The functional requirements for our application are shown in this use case diagram.

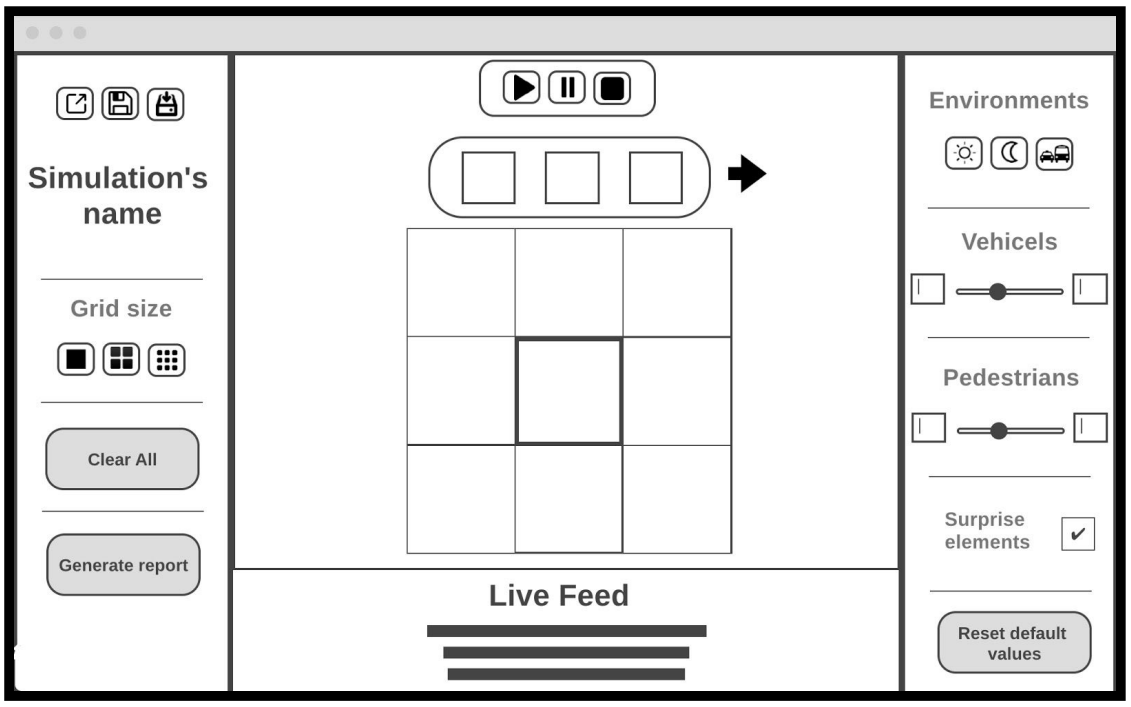
The must features will be explored further in the use cases section.

Use case Diagram



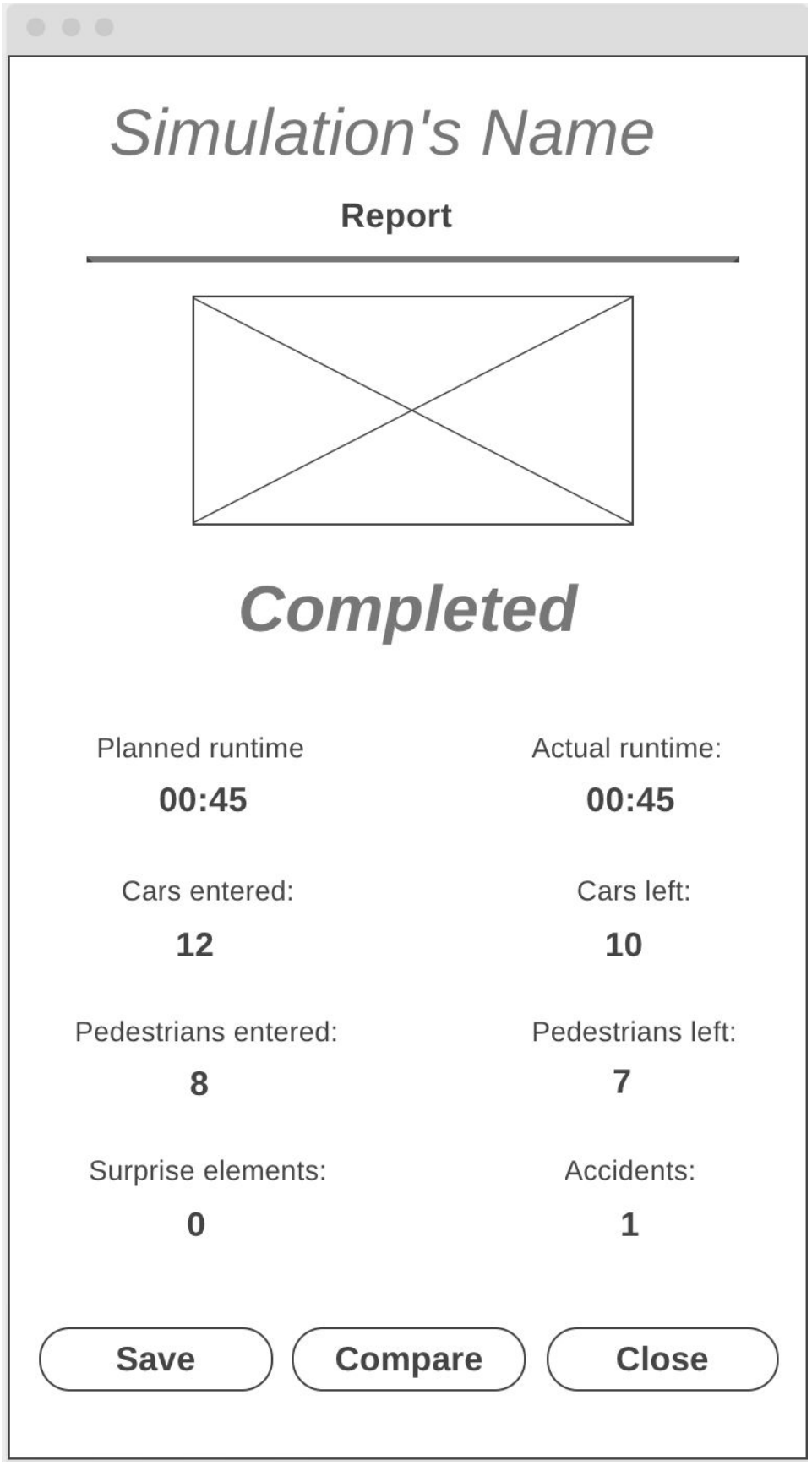
User Interface Sketches

Our application will have the following layout for the main screen:



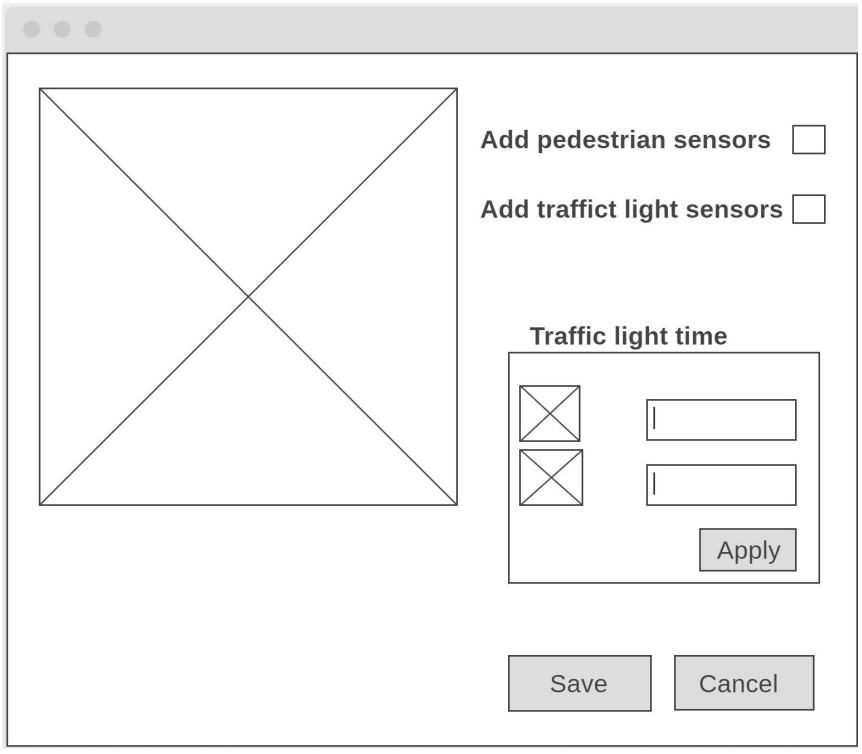
This window has 4 panels:

1- File & Grid options (Left panel): From top towards bottom you can see the file options (Open - Save - Save as). Underneath that you can find the simulation's name that the user can edit. Followed by grid size options (Small - Medium - Large). Next you can see a "Clear All" button which removes all elements from the grid. After that you can see the "Generate report" button which displays a report about the simulation that just took place, here's an example of such window:


Simulation's Name
Report
Completed
Planned runtime
00:45
Actual runtime:
00:45
Cars entered:
12
Cars left:
10
Pedestrians entered:
8
Pedestrians left:
7
Surprise elements:
0
Accidents:
1
Save
Compare
Close

2- Simulation's main screen (Top-center panel): This is the screen where the actual simulation is going to take place, on top you have "Play - Pause - Stop" buttons. Underneath you have the pieces list window, this window is visible whenever a user clicks on an empty piece of the grid. After that you have the grid on which the user can have the pieces on.

By double clicking on a specific piece, a window form for modifying the settings of the selected piece will be shown on the screen.



The user will be able to set the green and the red light time per traffic light and add / remove sensors. To change the green or the red time of a traffic light, the user will select the traffic light by clicking on it and inserting in the text boxes the desired amount of seconds for each traffic color.The settings are saved per traffic light by clicking the "Apply" button.

The user can also choose to add pedestrian or traffic light sensors.To save the changes,the user can click the "Save button" on the bottom of the form.The user will be informed if the changes have been successfully applied.

3- Live feed (Bottom-center panel): This window will provide updates/live feed about the simulation while it's taking place.

4- Dynamic elements (Right panel): This panel is for adding/removing/modifying live elements (pedestrians - vehicles - emergency vehicles) to the simulation. On top you can find Pre-set environments with 3 pre-defined environments (Normal day traffic - Normal night traffic - Rush hour), in which the user can click on one and it would modify the parameters to emulate that environment. Afterwards you have the pedestrians and vehicles sliders, which the user can use to specify how many cars/people this simulation is going to have. Next you have the "Surprise elements" checkbox, which adds a percentage of random events taking place on the map to simulate real-life traffic (e.g. An accident happening & ambulance/police cars showing up). Lastly we have the "Reset default values" button, which restores the default parameters of this panel. This panel is also unaccessible while the simulation is taking place.

USE CASES

In this section we're going to list all the required use cases that our application is going to have:

Start Simulation

ID: 1

Name: Start Simulation

Goal: Start the simulation

Actors: User

Description: The user clicks a button that starts the flow of cars hence the simulation

Pre-conditions: The user has constructed a road system by placing roads on the map

Trigger: Click of the button "Start Simulation"

Main Success Scenario:

1. User click button "Start Simulation"
2. System starts the simulation

Post-condition:

1. User can stop the simulation or exit
2. User can pause the simulation

Extensions:

2A: ***User restarts the program:***

- Use case ends

Stop Simulation

ID: 2

Name: Stop Simulation

Goal: Stop the simulation

Actors: User

Description: The user clicks a button that stops the simulation and clears the currently build road system

Pre-conditions: The simulation is running

Trigger: Clicking the button “Stop Simulation”

Main Success Scenario:

1. User clicks button “Stop Simulation”
2. System stops the simulation
3. System resets the road map

Post-condition:

1. The simulation is resetted.

Extensions:

-3A: ***User restarts the program:***

- Use case ends.

Pause Simulation

ID: 3

Name: Pause Simulation

Goal: Stop the simulation for a while/pause the simulation

Actors: User

Description: The user clicks a button that stops the simulation but does not end it

Pre-conditions: The simulation is running

Trigger: Clicking the button “Pause Simulation”

Main Success Scenario:

1. User clicks button “Pause Simulation”
2. System pauses the simulation without resetting the map

Post-condition:

1. User can continue the simulation or stop it

Extensions:

-2A: ***User restarts the program*:**

- Use case ends.

Generate a report

ID: 4

Name: Generate a report

Goal: Generate a report for the user of a simulation that already took place.

Actors: User

Description: The user is indicating to display a report that contains the stats of the simulation that just took place. The system displays the result.

Pre-condition:

1. A simulation has just taken place.
2. The simulation is stopped
3. The simulation had at least one dynamic element.

Trigger: Simulation has just ended, and the user indicates to display a report.

Main success scenario:

1. User indicates to display a report for the simulation.
2. System displays the report.

Post-condition:

- The report is shown, and the user can choose to continue.

Extensions:

-1A: ***The simulation had no dynamic elements (cars, pedestrians):***

- System displays an error message.

- The use case is over.

Reset Road Map Layout

ID: 5

Name: Reset road map layout

Goal: Remove all the elements off the grid.

Actors: User

Description: A user can choose to reset a layout displayed on the screen.

Pre-condition:

1. A map is already loaded on the grid.

2. The simulation is stopped.

Trigger: User wants to remove all elements off the grid.

Main success scenario:

1- User clicks on “Reset map”

2- System displays a verification message

3- User chooses to proceed.

4- All elements are removed off the grid.

Postcondition:

- Map grid is now empty.

Extensions:

-3A: ***User chooses to cancel:***

- Use case ends.

Add one-way road

ID: 6

Name: Add one-way road

Goal: Adding one-way road in the running application

Actors: User

Description: A user can choose to add one-way road to the grid

Pre-condition:

1. The Simulation is stopped.

Trigger: User indicates to add one-way road by clicking the button

Main success scenario:

1. Actor clicks add one-way button
2. Actor clicks on the grid
3. System add the road into the grid

Post-condition: The chosen road is added in the grid

Extensions: *None*

Remove one-way road

ID: 7

Name: Remove one-way road

Goal: Removing one-way road in the running application

Actors: User

Description: A user can choose to remove one-way road in the grid

Pre-condition:

1. There is at least one-way road on grid
2. The simulation is stopped.

Trigger: User wants to remove the one-way road by clicking the button

Main success scenario:

1. Actor clicks remove one-way button
2. Actor clicks on the grid
3. System remove the road into the grid

Post-condition: The clicked remove button Inform there is no road anymore if the road in the grid is empty

Extensions: *None*

Add two-way road

ID: 8

Name: Add two-way road

Goal: Adding two-way road in the running application

Actors: User

Description: A user can choose to add two-way road to the grid

Pre-condition:

1. The simulation is stopped.

Trigger: User indicates to add two-way road by clicking the button

Main success scenario:

1. Actor clicks add two-way button
2. Actor clicks on the grid
3. System add the road into the grid

Post-condition: The chosen road is added in the grid

Extensions: *None*

Remove two-way Road

ID: 9

Name: Remove two-way road

Goal: Removing one-way road in the running application

Actors: User

Description: A user can choose to remove two-way road in the grid

Pre-condition:

1. There is at least two-way road on grid
2. Simulation is stopped

Trigger: User wants to remove the two-way road by clicking the button

Main success scenario:

1. Actor clicks remove two-way road button
2. Actor clicks on the grid
3. System remove the road into the grid

Post-condition: The clicked remove button Inform there is no road anymore if the road in the grid is empty
Extensions: *None*

Add intersection

ID: 10

Name: Add intersection

Goal: Add an intersection to the map on the grid.

Actor: User

Description: A user can choose to place an intersection on an empty space on the grid.

Pre-condition: The software is running.

Trigger: User wants to add an intersection to the grid.

Main success scenario:

1. System shows the available intersections.

2. User chooses one and clicks on it.
3. User clicks on the grid where he wants to put the intersection.
4. System draws the intersection on the grid.

Post-condition: an intersection is placed on grid.

Extensions:

-3A: ***The place on the map is already taken by another object:***

- the system triggers an error sound.

- Nothing changes on the map.

- Back to step 1.

Remove intersection

ID: 11

Name: Remove intersection

Goal: Remove an intersection from the current map.

Actor: User

Pre-condition: The software should be running

Trigger: User wants to remove an intersection from the current map

Main success scenario:

1. User clicks on the intersections that he wants to remove
2. User chooses from the menu the remove option
3. System removes the selected intersection

Post-condition: Intersection is removed from the screen.

Extensions: *None*

Add Road with traffic light & Zebra crossing

ID: 11

Name: Add road with a traffic light & zebra crossing

Actor: User

Goal: To add a road with a traffic light to the current map.

Pre-condition: The Software should be running

Trigger: User wants to add a road with a traffic light & zebra crossing to the map

Main success scenario:

1. System shows the available roads with traffic lights & zebra crossings
2. Actor chooses one and clicks on it
3. Actor click on the map where he wants to put the road with traffic lights & zebra crossings
4. System draws the road with traffic lights & zebra crossings on the map

Post-condition: Road with traffic lights & zebra crossings is drawn on screen.

Extensions:

-3A: ***If the place on the map is already taken by another object****:*

- the system triggers an error sound

- Map stays unchanged..

- Back to step 1.

Remove road with traffic light & zebra crossing

ID: 12

Name: Remove road with traffic light & zebra crossing

Actor: User

Goal: To remove a road which has a traffic light & zebra crossing

Pre-condition: The Software should be running

Trigger: User wants to remove a road with a traffic light & zebra crossing

Main success scenario:

1. Actor clicks on the road with traffic lights & zebra crossings that he wants to remove
2. Actor chooses from the menu the remove option
3. System removes the selected road with traffic lights & zebra crossings

Post-condition: Road with traffic lights & zebra crossings is removed from the screen.

Extensions: *None*

Save a file

ID : 13

Name: Save a file

Goal: Save the contents of the currently open application

Actor: User

Trigger: User chooses option “Save”

Main Success Scenario:

1. The application saves the contents using the current name and location of the file.
2. The system displays the time and date of the last save in a message.

Post-condition: The content of the application is stored in a file.

Extensions:

-1A: ***Current application has not been given a name and location***

1. System goes to “Save As” use case

Save as a file

ID: 14

Name: Save As file

Goal: Saving contents of currently open file

Actor: User

Trigger:

- User chooses option “Save As”

- User tried to perform “Save a file” use case but did not specify location nor name for the file.

Main Success Scenario:

1. Application asks for name and location of file to be saved
2. Actor provides name and location of file
3. Actor confirms by clicking the ”Save” button
4. Application saves contents using given name and location of file

Extensions:

-2A: ***Actor presses “Cancel” button:***

1. Use case ends

-3A: ***There is already a traffic file with the same name and path:***

1. The system displays appropriate messages and offers choices to replace existing files.
2. If yes, the use case continues.
3. If not, the actor is returned to step 4.

Specify number of cars generated

ID: 15

Name: Specify number of cars generated

Goal: Specify the number of cars that will take part in the traffic flow

Actor: User

Trigger: User signals to change the default value of cars generated

Main Success Scenario:

1. User types in text box the desired number of cars to be shown

Extensions:

-1A: ***The value entered is not a number type/ exceeds the maximum value allowed/ is not a positive value:***

- The system shows an appropriate message.

- Return to step 1 of MSS.