User Requirements Specification

Final Version



Group B

Todor Tsekov (2228777)

Wen Zhang (2351420)

Jiaqi Ni (2217774)

Yidi Wu (2197301)

Kalina Petrova (2221667)

Jianfei Feng (2204529)

Class: EI7s1/s2

Course: Project C-phase

Mentor: Bert Van Gestel

Table of Contents

1. *Introduction------------------------------------------------------------------------3*
2. *Functional Requirements-------------------------------------------------------3*
3. *User Interface---------------------------------------------------------------------12*
4. *Non- functional Requirements------------------------------------------------14*

Introduction

In this project, we have to build traffic-simulation program and the first step is to write the User Requirements Specifications. For this step we will provide the use-cases of all functionality that we can offer, a specification of user interface and also non-functional requirements. The objective of our application is to have less traffic jams.

Our application is supposed to help users to create traffic simulation and handle the traffic. It will allow the users to build simulation with crossings. The user will be able to put crossings in the grid, and the user can watch the cars flow in the street. The user can add start point and destination to see how the car flows in the fast and easy way. The user can set traffic light in the crossing and position options of lights will be provided.

Functional Requirements

The following use cases will show the actions the user can perform within the application. Based on that, we assume the user can create new grids, load existing ones, and manipulate crossings, traffic lights and flow of traffic.

# MoSCoW Table:

|  |  |
| --- | --- |
| Use-cases: | Implementation: |
| Add Crossing | Must |
| Delete Crossing | Must |
| Change Crossing | Should |
| Rotate Crossing | Should |
| Modify Traffic Light System | Should |
| Change Traffic Light Setup | Could |
| Alter Flow | Should |
| Navigate | Could |
| Play Simulation | Must |
| Pause Simulation | Must |
| Stop Simulation | Must |
| Create New Project | Must |
| Load Project and Statistics | Must |
| Save Project and Statistics | Must |
| Exit Application | Must |
| Go to Main Menu | Should |
| Undo | Could |
| Redo | Could |
| Reset | Could |

**States:**

**Initial state**- where you build the simulation (crossings,etc.)

**Simulation running state**- when you test the simulation

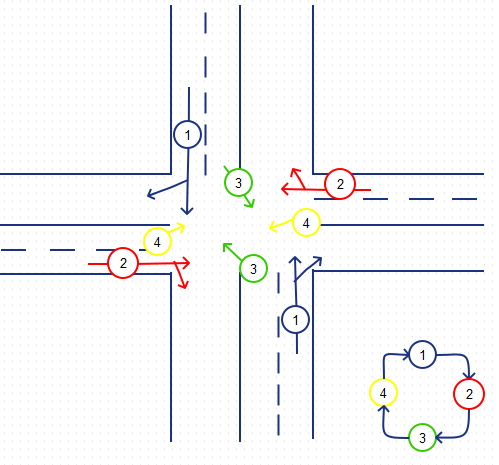
**Pause state**-where you stop to see the progress of the simulation till this point

**Traffic light states:**

When we create a crossing, we will automatically create a traffic light system. Each lane of the crossing will have its own traffic light. We will combine traffic lights into groups or states as we call them. Each group will be green at a specific time of the cycle. The user will be able to change only the green time of a specified cycle and not a traffic light by itself. That way we make sure that the user doesn’t make a mistake with his inputs. Refer to the Figure bellow for an example of a traffic light system setup.

**“Change traffic light system setup” specifications:**

When a user wants to change the type of a traffic light system he’d like to use on the specified crossing. For now we have limited the user to a set of traffic light setups for different crossings. Later in the implementation we may be able to let the user create his own groups of traffic lights and create his own states.



So as you can see lanes numbered 1 will go together then they’ll switch with lanes numbered 2 and so on.

# Use-cases:

1. **Name**: Add crossing

**Goal**: Adds a crossing to our grid

**Actor**: User

**Pre-condition**: System is displaying a new or a loaded project so that the user can modify the grid. System is in state “initial state”.

**MSS**:

1. User chooses a crossing he wants to add
2. User drags a crossing to the cell he wants to place it in
3. User drops the crossing in the cell
4. System redirects user to traffic light system setup
5. System asks user what kind of traffic light system he would like to choose(Optional)
6. User selects a traffic light system(Optional)
7. System asks user to input values of the traffic light system
8. User input values for each traffic light system state
9. System places the crossing on the grid
10. System adds new cell to the calculation of the flow

**Exception (Extension, Alternatives)**:

3.1: Cell already has a crossing on it

System asks user whether he’d like to change the crossing on the cell.

User clicks ok -> Go to “Change crossing Use-case”

User clicks cancel -> The system doesn’t modify the grid in any way

9.1: User doesn’t give inputs for traffic light system

System configures the traffic light system with default values

9.2: User inputs wrong values

System outputs an appropriate message “You have to input values of type integer 1-100”

9.3: User sets the time to too high or too low

System notifies the user of the boundaries

**Post-condition**: Our grid now has a new crossing on it.

1. **Name**: Delete crossing

**Goal**: Deletes an existing crossing from the grid

**Actor**: User

**Pre-condition**: System is displaying a new or a loaded project so that the user can modify the grid and there’s already a crossing on the grid to delete. System is in state “initial state”.

**MSS**:

1. User right clicks on existing crossing on the map
2. User chooses the delete option
3. System asks the user for confirmation
4. The user confirms
5. System deletes crossing from the grid
6. System removes the cell from the calculation of the flow

**Exception (Extension, Alternatives)**:

1.1: When a user right clicks on an empty cell, the delete crossing option won’t appear

5.1: Crossing was part of the path from start point to end point

System deletes navigation route

**Post-condition**: Our grid now is operational without the specified crossing that we deleted.

1. **Name**: Change crossing

**Goal**: Changes an existing crossing on the grid

**Actor**: User

**Pre-condition**: System is displaying a new or a loaded project so that the user can modify the grid and there’s already a crossing on the grid to change. System is in state “initial state”.

**MSS**:

1. User right clicks on a cell with a crossing
2. User chooses the change option
3. System redirects user to crossing settings
4. User choose crossing he wants to place in said cell
5. System asks user to input values of the traffic light system
6. User input values for each traffic light system state
7. System removes previous crossing from the grid
8. System places the new crossing in the cell
9. System automatically changes the flow of the traffic

**Exception (Extension, Alternatives)**:

1.1: When a user right clicks on an empty cell, the change crossing option won’t appear

**Post-condition**: The grid now has a different kind of layout- there’s now a new crossing in place of our previous specified one.

1. **Name**: Rotate crossing

**Goal**: Rotates an existing crossing on the map

**Actor**: User

**Pre-condition**: System is displaying a new or a loaded project so that the user can modify the map and there’s already a crossing on the map to rotate.

**MSS**:

1. User right clicks on existing crossing
2. User chooses the rotate option
3. System swaps the properties of the 4 roads with the properties of the one on the right.
4. System rotates the crossing picture(90 degrees counter clockwise)
5. System automatically recalculates the flows

**Exception (Extension, Alternatives)**:

**Post-condition**: The map has now a different kind of layout- our specified crossing has been rotated and the system has recalculated flows accordingly.

1. **Name**: Modify traffic light system

**Goal**: Modify an existing traffic light system on the grid

**Actor**: User

**Pre-condition**: System is displaying a new or a loaded project so that the user can modify the grid and there must already be a crossing with a traffic light. System is in state “initial state”.

**MSS**:

1. User hovers over a cell
2. System output a small icon in the top left corner
3. User clicks on the icon
4. System redirects user to traffic light and flow settings
5. User chooses to modify the interval times of the states of the traffic light system
6. System shows the user the current parameters
7. User changes the values of the inputs of the states
8. User clicks confirm
9. System modifies traffic light system with new parameters
10. System recalculates total interval time

**Exception (Extension, Alternatives)**:

7.1: User leaves input value blank

System modifies the traffic light with default value

7.2: User inputs wrong values

System outputs an appropriate message “You have to input values of type integer 1-100”

7.3: User sets the time to too high or too low

System notifies the user of the boundaries.

**Post-condition**: The selected traffic light has now changed parameters.

1. **Name**: Change traffic light setup(Optional)

**Goal**: Changes an existing traffic light system on the grid to another one

**Actor**: User

**Pre-condition**: System is displaying a new or a loaded project so that the user can modify the grid and there must already be a crossing with a traffic light. System is in state “initial state”.

**MSS**:

1. User right clicks on a crossing
2. User chooses the “change traffic light system” option
3. System shows the user the traffic light setups the user can choose from
4. User selects a new set up for the traffic light system
5. System asks user to input values of states
6. User inputs values for each state
7. System replaces existing traffic light system with a new one

**Exception (Extension, Alternatives)**:

6.1: User leaves input value blank

System modifies the traffic light with default value

6.2: User inputs wrong values

System outputs an appropriate message “You have to input values of type integer 1-100”

6.3: User sets the time to too high or too low

System notifies the user of the boundaries.

**Post-condition**: The selected traffic light system has now been changed to another.

1. **Name:** Alter flow for crossing

**Goal:** Alter flow for existing crossing

**Actor:** User

**Pre-condition:** System is displaying a new or a loaded project so that the user can modify the grid and there’s already at least one crossing on the map to edit. Available incoming lane (border lane) for modifying flow will show in “Crossing Setting” window. System is in state “initial state”.

**MSS:**

1. User right clicks on the existing crossing

2. User chooses “Crossing Setting”

3. User modify car flow for each incoming lane.

4. User click “confirm” button.

**Exception (Extension, Alternatives):**

2.1: If user do not change anything for flow, then it will remain default value.

Post-condition: none.

1. **Name:** Navigate

**Actor:** User

**Precondition:** System is in state “initial state”.

**MSS:**

1. User chooses “navigation” option.
2. User sets the starting point on the screen.
3. User sets the destination point on the screen.
4. User confirms the starting point and destination point.
5. User input the flow number.
6. System calculates the route and executes the “Alter flow” use case with the input flow number for the related roads.
7. System goes to state “Simulation running state”.

**Extension:**

4.1 User cancels the operation.

**Post-condition:** The system simulates the situation and changes the flow of the related roads correctly.

1. **Name:** Play Simulation

**Goal:** The system simulates the project.

**Actor:** User

**Pre-condition:** Actor is at the “Project-grid screen”. System is in state “paused state” or “initial state”.

**MSS:**

1. Actor click play button.

2. System starts the simulation (the play button is changed with a pause button-so you can pause the simulation).

**Exception:**

1.1 The play button will be disabled for rules 9 and 10 (see the rule section).

1. **Name:** Pause Simulation

**Goal:** Pause the system

**Actor:** User

**Pre-condition:** The actor is at “Project-grid screen”. System is in state “Simulation running state”.

**MSS:**

1. Actor clicks the Pause button.

2. The system is paused and it stays in a “pause state” (the pause button is change with a play button-so you can start the simulation).

**Exception:** None

1. **Name:** Stop Simulation

**Goal:** Stop the system

**Actor:** User

**Pre-condition:** The actor is at “Project-grid screen”. System is in state “Simulation running state” or “paused state”.

**MSS:**

1. Actor click stop button.

2. System stop the simulation and we can work on the “Project-grid screen” again (the system is in “initial state”).

**Exception:** None

1. **Name:** Create new project

**Goal:** Create new project

**Actor:** User

**Pre-condition:** The actor is at “Main screen”

**MSS:**

1. Actor click new simulation button.

2. The system creates a new project and the user sees the Project-Grid screen (the system changes it status to initial state).

**Exception:** None

1. **Name**: Load project and statistics

**Goal**: Load a Project

**Actor**: User

**Pre-condition**: System is displaying the application “Main Screen”.

**MSS**:

1. User clicks the “Load” button;
2. A file-dialog window will pop-up and user chooses the file which is going to use;
3. User selects a file and clicks “Ok”;
4. System loads the file and displays the Project-Grid screen(the system is in “initial state”);

**Exception (Extension, Alternatives)**:

3 – a) If there is a project running, the user can not choose a new file. It shows a message” The has existed a project ”.

3 – b) User may want to change a file, the user needs to close an existing file first(is not possible to open two files at the same time) and goes to use case 15(Exit Application).

Now user can select a new file.

4 – a) If the system doesn’t have enough permissions to open the file, it displays a message “Not enough permissions to open this file” and the use case ends.

4 – b) If the system can’t parse the file correctly, it displays a message “This file could not be loaded.” and the use case ends.

**Post-condition**: The project-grid screen will be ready for the user.

1. **Name**: Save project and statistics

**Goal**: Save to a file

**Actor**: User

**Pre-condition**: System is displaying project- grid screen. System is in state “initial state”.

**MSS**:

1. User chooses “File-Save” option, on the top left corner;
2. System saves the current grid to a file;

**Exception (Extension, Alternatives):**

2 – a) If the current grid hasn’t been saved before, the system displays a file-dialog where the user needs to choose the folder and the name of the file he wants to save.

2 – b) If the system doesn’t have permissions to save the file, it displays a message “Not enough permissions to save this file” and the use case ends.

**Post-condition**: The file is saved by the system.

**15.Name**: Exit Application

**Goal**: Close a file

**Actor**: User

**Pre-condition**: System is displaying project-grid screen. System is in state “initial state”.

**MSS**:

1. User chooses “File-Close” option, on the top left corner;
2. System asks the user for confirmation;
3. The system exits;

**Exception (Extension, Alternatives):**

2 – a) User clicks ’No’ option, the system will not exit.

2 – b) There are files that have been edited without being saved.

The system asks if you want to save opened files.

User may close file without saving.

2 – c) User may want to save file before close it.

Then user goes to use case 14 (Save project and statistics).

**Post-condition**: The file is closed by the system.

1. **Name**: Go to Main Screen

**Goal**: Takes the user back to the main screen

**Actor**: User

**Pre-condition**: System is displaying project-grid screen. System is in state “initial state”.

**MSS**:

1. User chooses “File- main menu” option, on the top left corner;
2. System closes project- grid screen and opens main screen(the system is in “working state”);

**Exception (Extension, Alternatives):**

2 – a) If the user didn’t save the progress, the system will show a message asking if the user wants to leave without saving. The user can choose “Yes” or “No”.

**Post-condition**: Main screen is displayed.

1. **Name**: Undo

**Goal**: Undo a step

**Actor**: User

**Pre-condition**: System is displaying project-grid screen and the user did at least one step (add crossing, change flow, etc.) System is in state “initial state”.

**MSS**:

1. User clicks the “Undo” button;
2. The system shows grid before the user’s last action on the screen;

**Post-condition**: The grid is the way it was before the last step.

1. **Name**: Redo

**Goal**: Redo a step

**Actor**: User

**Pre-condition**: System is displaying project- grid screen and the user did at least one undo action. System is in state “initial state”.

**MSS**:

1. User clicks the “Redo” button;
2. The system shows grid after the user’s last action on the screen;

**Post-condition**: The grid is the way it was after the last undo.

1. **Name**: Reset

**Goal**: Clear the current grid

**Actor**: User

**Pre-condition**: System is displaying project-grid screen. System is in state “initial state”.

**MSS**:

1. User chooses the “File- Reset” option;
2. System displays a message, asking the user to confirm the action;
3. User confirms the action;
4. The system shows the empty grid on the screen;

**Exception (Extension, Alternatives)**:

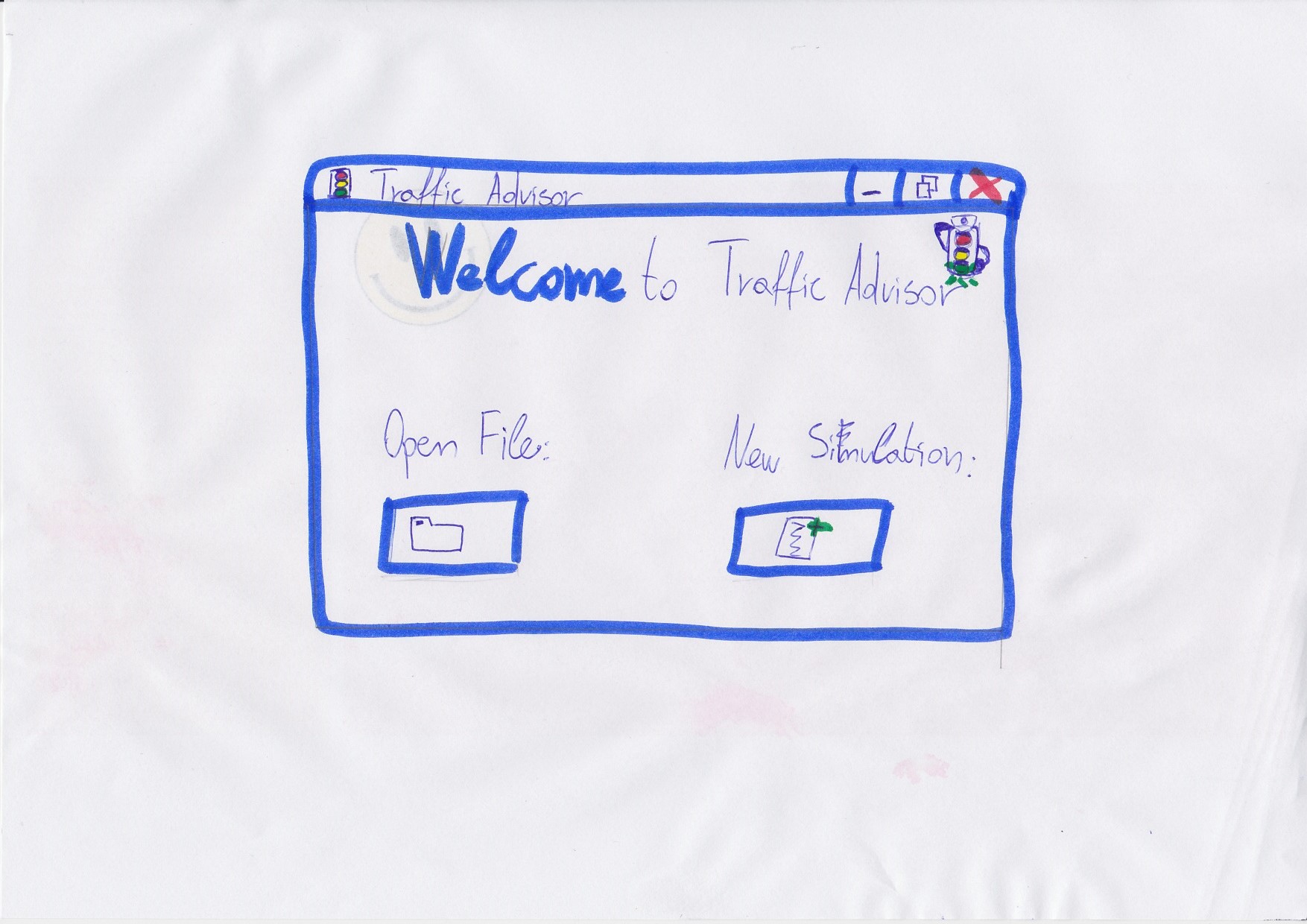
3 – a) The user clicks in the “No” option, the system display grid the way it was and the use case ends.

**Post-condition**: The grid is empty.

User Interface

We created a storyboard of the user interface. These are the two screens that we are going to have:

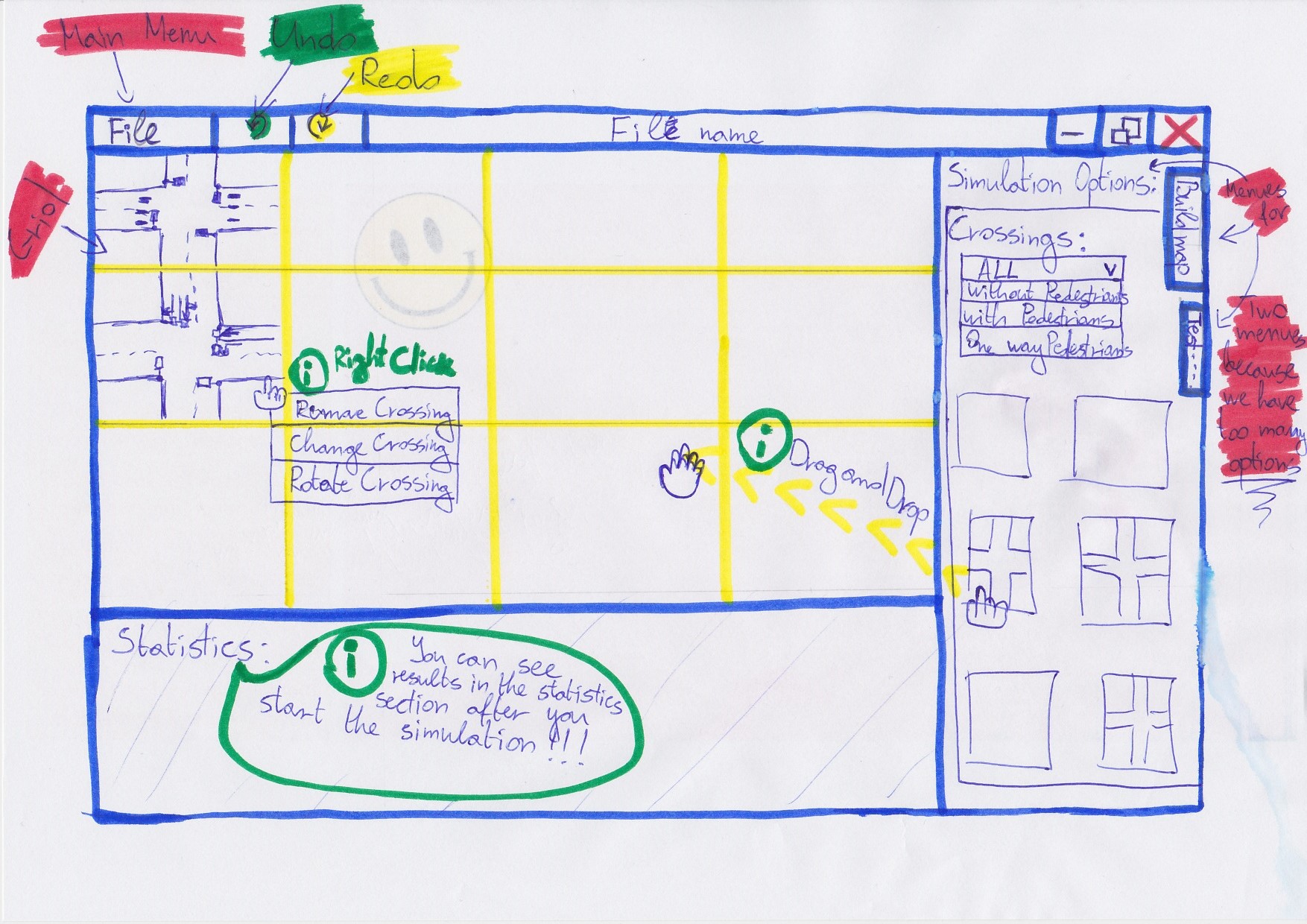
Main screen:



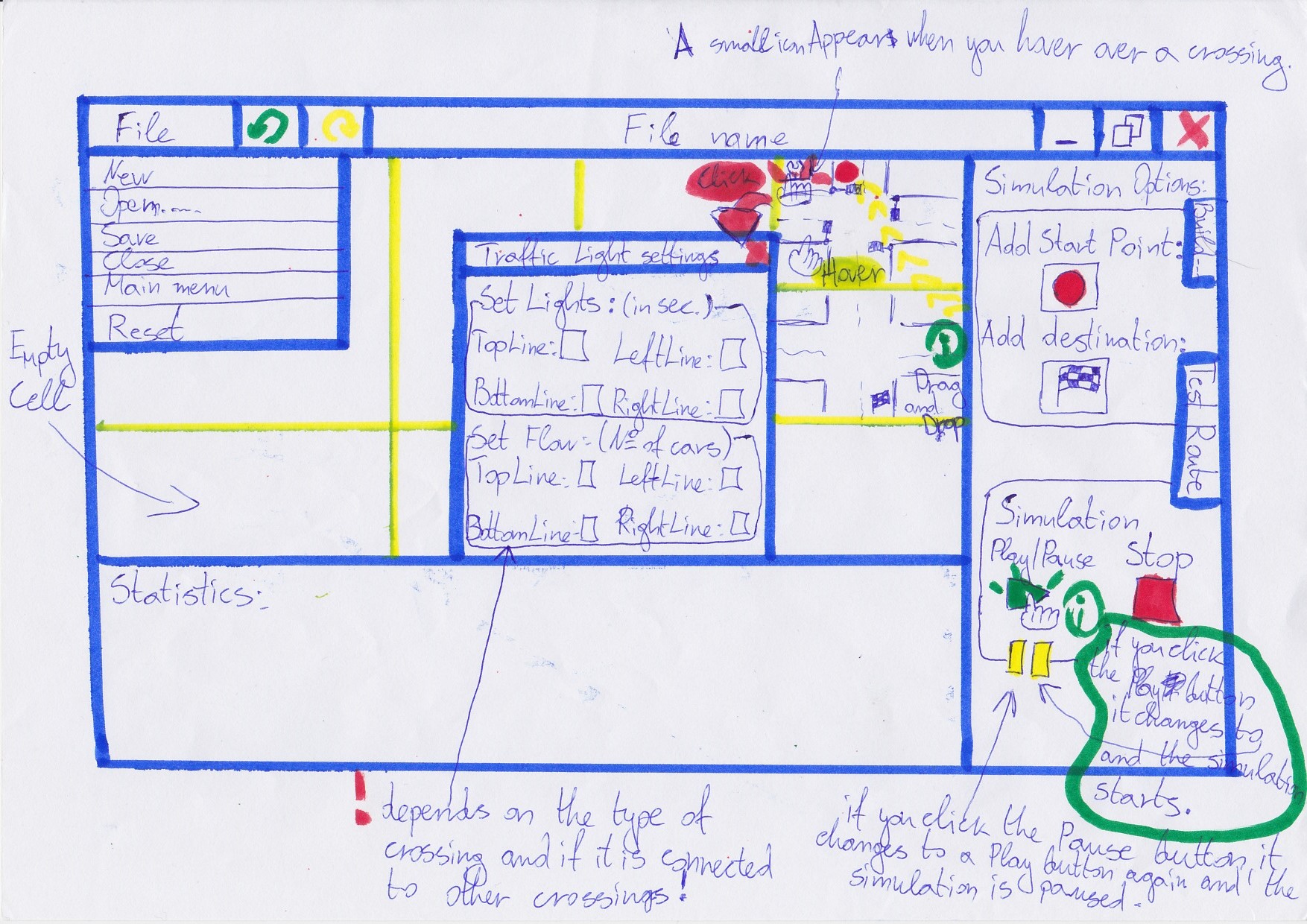
With ‘Open File’ button you can open a simulation file from your PC.

With ‘New Simulation’ button you can start new simulation on your PC.

Project-grid screen:



On this screen is the main functionality of the application, we could choose from a lot of options, so we made another look of this form so you could see the rest of the options.

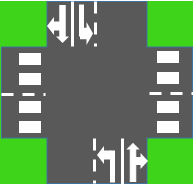
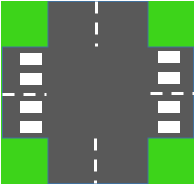


State 1:\* State 2:

State 3:\* State 4:

# Crossings:

This is our proposal for suitable crossing for the traffic advisor application.



# Rules:

1. When a Cross is rotated all its data will be reset to default options.
2. You cannot change any traffic or crossing settings in Simulation running state. Disable interactive icons when in state “playing”, and enable them back on pause.
3. A car will go throw the crossing for 3 seconds. It cannot be changed.
4. The yellow light of a traffic light will be 3 seconds. It cannot be changed.
5. If a crossing is connected with other crossings and is part of the calculations for the Start point and Destination point, if we remove it from the grid these points will be removed from the grid as well (you need to choose new points or add a crossing).
6. You cannot input more than x seconds and less than y seconds on the traffic lights.(x and y are the limits that we are going to set for a traffic light)
7. If there is a pedestrian crossing on a Cross the system counts that each pedestrian crossing has a Green Light for 30 seconds.
8. Results in the statistics section will appear on the screen after the simulation is started.
9. Simulation will not start if a ‘Start Point’ and ‘Destination’ points are not chosen.(the system cannot calculate the traffic statistics without it)
10. Simulation will not start if there are not at least two Crosses on the grid, because to calculate the statistics for the star point and destination point they need to be on separate crossings.

Non-functional Requirements

* The application will be programmed in C#, which means Windows operation system version windows 7 or windows 8 are the required environments for it.
* The application can run on a basic computer.
* You do not need Internet to use our application.