

**Author information**

Name: Lucassen, Mario M.  
Course: Software Engineering  
Email: M.lucassen@student.fontys.nl

---

# **PAZURU SUDOKU PUZZLE SPECIFICATION**

---

Document no: 2  
Version: 1.00  
File: Pazuru Sudoku Puzzle  
Specification.docx

Created: 10.05.2019  
Last changes: 10.05.2019

## Version history

Version	Date	Changes	Author <sup>1</sup>
0.01	10.05.2019	Document created with Introduction, Actions and Rules.	Lucassen, Mario M. M.lucassen@student.fontys.nl
1.00	10.05.2019	Final version of document.	Lucassen, Mario M. M.lucassen@student.fontys.nl

---

<sup>1</sup> Author's name with email address

# Table of contents

<b>1</b>	<b>Introduction .....</b>	<b>4</b>
1.1	Purpose .....	4
1.2	Scope .....	4
1.3	Author Profile .....	4
<b>2</b>	<b>Requirements .....</b>	<b>5</b>
2.1	Actions.....	5
2.2	Rules .....	5
2.3	Quality attributes .....	5
<b>3</b>	<b>Use cases .....</b>	<b>6</b>
3.1	Use case diagrams.....	6
3.2	Use cases .....	7
3.2.1	Add number .....	7
3.2.2	Remove number .....	13
3.2.3	Solve Sudoku.....	17
3.2.4	Generate Sudoku .....	19
<b>4</b>	<b>User interface specification .....</b>	<b>21</b>
4.1	Home page (Sudoku generated) .....	21
<b>5</b>	<b>Solving algorithm .....</b>	<b>22</b>
<b>6</b>	<b>Generating algorithm .....</b>	<b>23</b>
<b>7</b>	<b>Puzzle state representation .....</b>	<b>24</b>

Table 1. Global definitions.

Name	Description
Fontys	Fontys University of Applied Sciences.

# 1 Introduction

## 1.1 Purpose

This specification document is made because each puzzle in the Pazuru platform needs its own specification document with the rules and actions explained.

A Sudoku puzzle is defined as a logic-based, number-placement puzzle. The objective is to fill a 9x9 grid with digits in such a way that each column, each row, and each of the nine 3x3 grids, that make up the larger 9x9 grid, contains all of the digits from 1 to 9. Each Sudoku puzzle begins with some cells filled in. The player uses these seed numbers as a launching point toward finding the unique solution. A solver for the puzzle will be made using a backtracking-algorithm. The puzzle generator will be made using a naive-algorithm.

## 1.2 Scope

The scope of the product being made is a subsystem of the Pazuru puzzle platform. The scope also establishes boundaries of the requirements and identify features/requirements outside and inside of the scope.

### Scope includes

Requirements that are listed with an importance level of 'Must', 'Should' or 'Could' in chapter 2, [Requirements](#).

### Scope excludes

Due to time constraints, not all requirements will be made. Requirements that are listed with an importance level of 'Wont' will not be included in the release product.

## 1.3 Author Profile

The author of the Sudoku puzzle implementation is Lucassen, Mario M, a Software Engineering student at Fontys. The product will be developed at the Fontys University of Applied Sciences, Rachelsmolen 1 in Eindhoven during the time allotted for the Software Engineering course.

### Author information

Name:	Lucassen, Mario M.	Course:	Software Engineering
Email:	M.lucassen@student.fontys.nl	Role:	Software Developer

## 2 Requirements

The requirements have been composed using the MoSCoW method which can be found in the tables below, under the heading Importance. The actions and rules described here are focused on the logic needed for the puzzle.

### 2.1 Actions

The actions for the system are shown in a table with per actions a name, description, importance, urgency and use case ID.

ID	Name	Description	Importance	Urgency	UC ID
F1	Add number	To add a number to the grid.	Must	High	UC11
F2	Remove number	To remove a non-seeded number from the grid.	Must	High	UC12
F3	Solve puzzle	To solve a Sudoku puzzle.	Must	High	UC13
F4	Generate puzzle	To generate a Sudoku puzzle.	Must	High	UC14

### 2.2 Rules

The rules for the system are shown in the table with per rule a description, importance, and urgency.

ID	Description	Importance	Urgency
R1	Each row can only contain each number from 1 to 9 once.	Must	High
R2	Each column can only contain each number from 1 to 9 once.	Must	High
R3	Each 3x3 sub-grid can only contain each number from 1 to 9 once.	Must	High
R4	A sudoku puzzle may only have 1 unique solution.	Must	High

### 2.3 Quality attributes

The quality attributes for the system are shown in the table with categories used by the ISO 25010<sup>2</sup>.

ID	Category ISO 25010	Description
Q1	Usability	The system can be used by anyone within 10 minutes.
Q2	Maintainability	The source code is written according to the Microsoft code conventions <sup>3</sup> for C#
Q3	Maintainability	Source code is documented.
Q4	Maintainability	At least 80% of the source code is covered by tests.

---

<sup>2</sup> ISO 25010:

<https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>

<sup>3</sup> C# Microsoft Code Conventions:

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions>

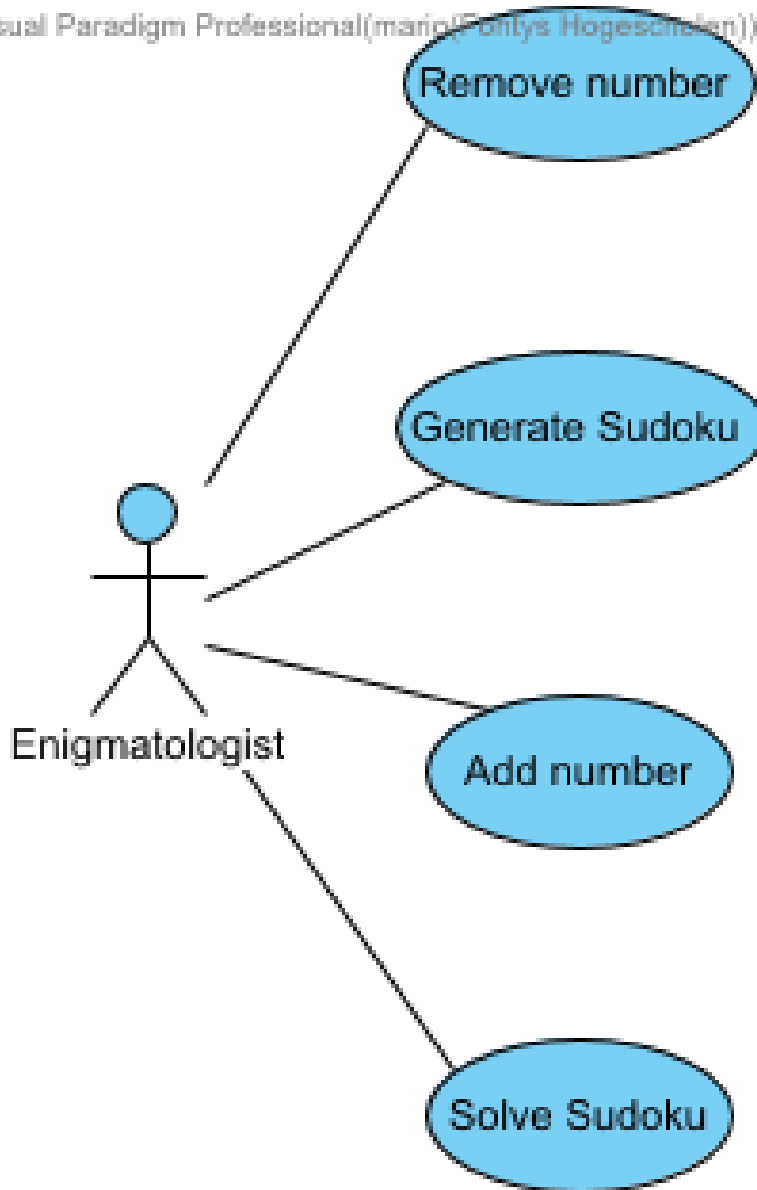
### 3 Use cases

This chapter contains Use case diagram(s) and Use cases for the platform. All the Use case diagram(s) & Use cases are created with Visual Paradigm using the UML<sup>4</sup> specification standard.

#### 3.1 Use case diagrams

Pazuru Sudoku Use case diagram.

Visual Paradigm Professional(mario(Fortys Hogescholen))



---


<sup>4</sup> Unified Modeling Language Specification 2.5.1:  
<https://www.omg.org/spec/UML/2.5.1>

## 3.2 Use cases

Pazuru Sudoku Use cases.

### 3.2.1 Add number

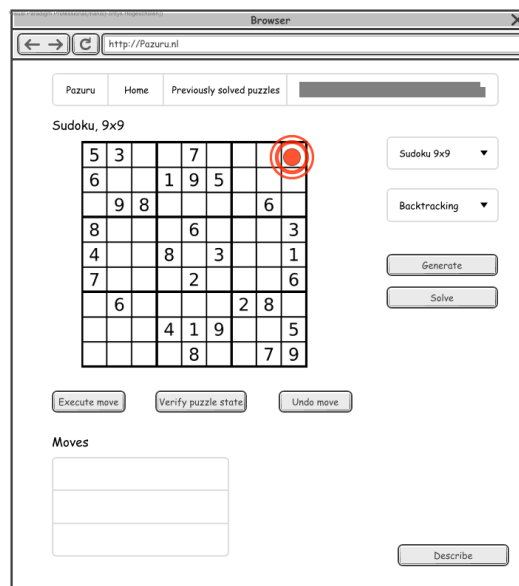
#### Add number

ID	UC11
Description	To add a number to the grid.
Primary Actors	 Enigmatologist
Level	User
Complexity	Medium
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	Generate Sudoku has executed successfully.
Post-conditions	The ACTOR has placed a number in the puzzle grid.
Author	Mario Lucassen
Assumptions	N/A

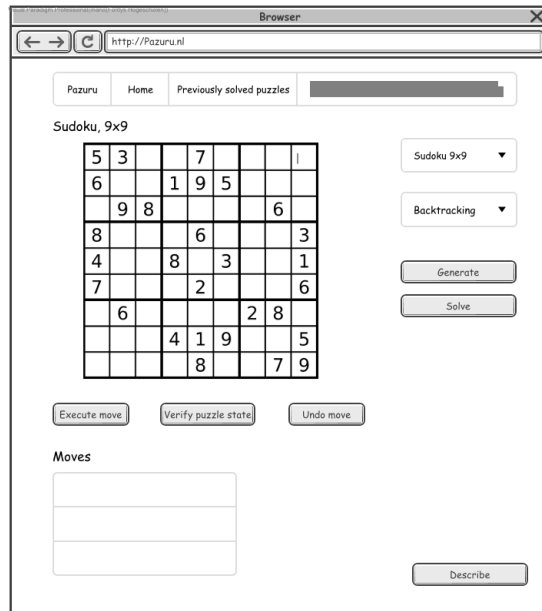
#### 3.2.1.1 Scenarios

##### Happy path

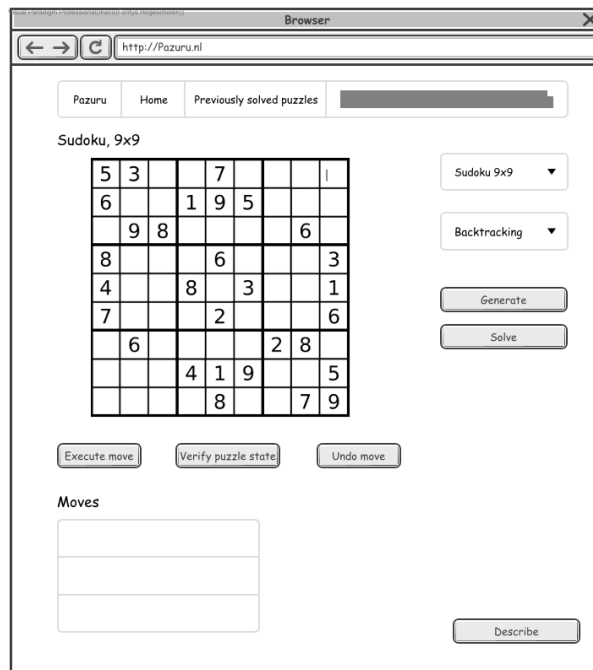
1. The ACTOR clicks on right most and top most cell.
2. **SYSTEM** Displays a caret in the designated cell.
3. The ACTOR presses the number 4.
4. **SYSTEM** Displays the number 4.
5. The ACTOR clicks on the **Execute move** button.
6. **SYSTEM** Performs the move and adds the move to the moves list.



1. The ACTOR clicks on right most and top most cell.

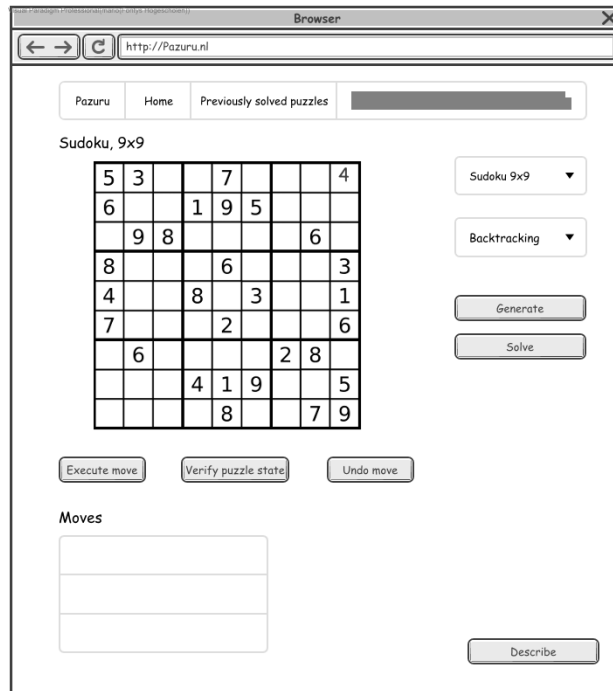


2. SYSTEM Displays a caret in the designated cell.

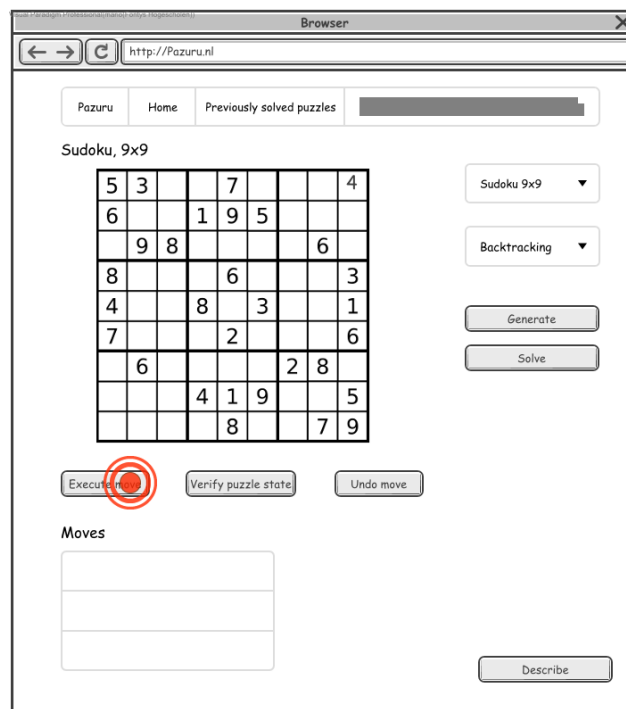


3. The ACTOR presses the number 4.





4. SYSTEM Displays the number 4.



5. The ACTOR clicks on the Execute move button.

Pazuru
Home
Previously solved puzzles

Sudoku, 9x9

5	3			7				4
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Sudoku 9x9

Backtracking

Generate

Solve

Execute move

Verify puzzle state

Undo move

Moves

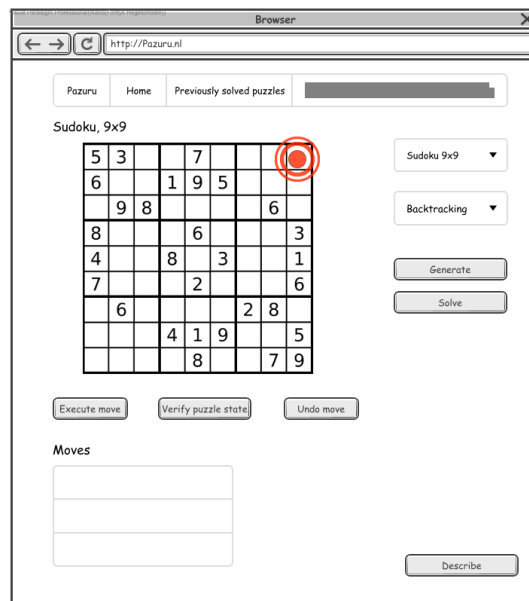
Row 1, Column 9, Number 4: ADD

Describe

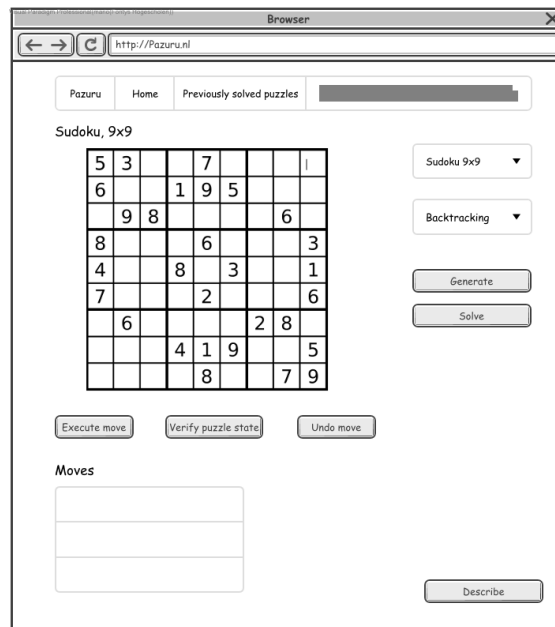
6. SYSTEM Performs the move and adds the move to the moves list.

### Invalid input path

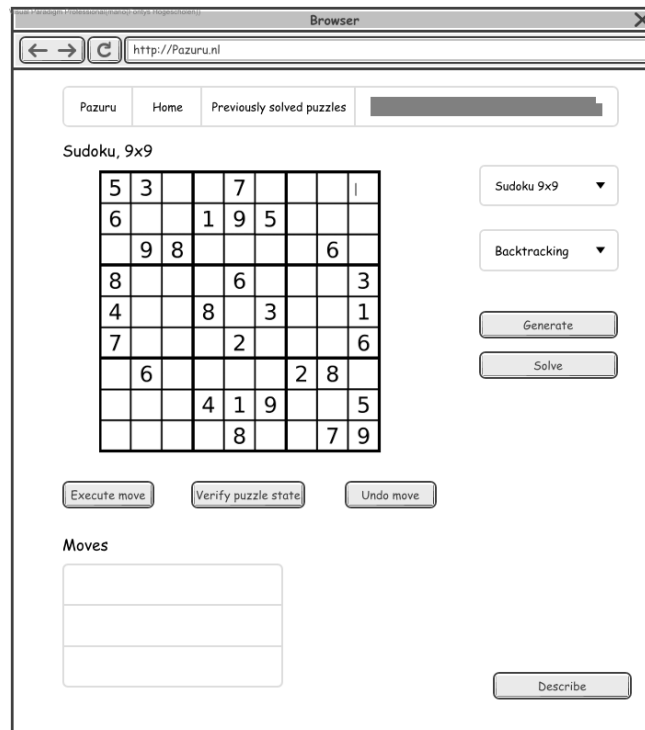
1. The ACTOR clicks on right most and top most cell.
2. **SYSTEM** Displays a caret in the designated cell.
3. The ACTOR presses the alphabetical character A.
4. **SYSTEM** Displays an Invalid input error popup message.



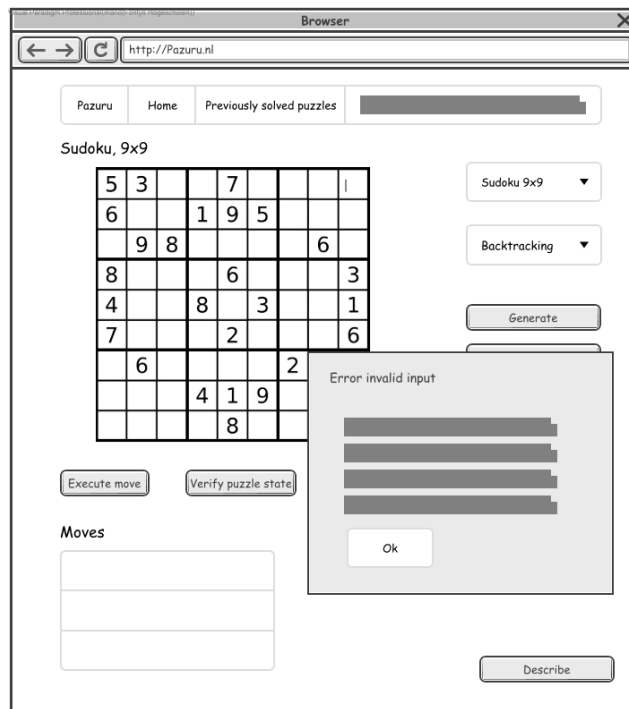
1. The ACTOR clicks on right most and top most cell.



2. **SYSTEM** Displays a caret in the designated cell.




3. The ACTOR presses the alphabetical character A.



4. SYSTEM Displays an Invalid input error popup message.

### 3.2.2 Remove number

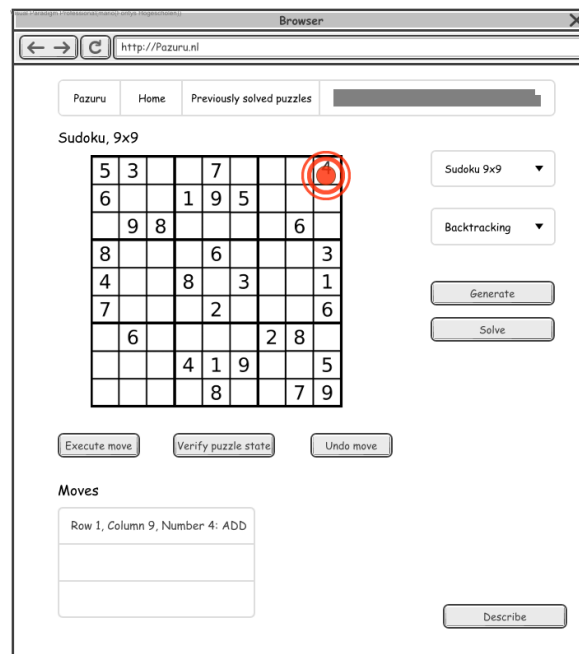
#### Remove number

<b>ID</b>	UC12
<b>Description</b>	To remove a non-seeded number from the grid.
<b>Primary Actors</b>	 Enigmatologist
<b>Level</b>	User
<b>Complexity</b>	Medium
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Scheduled
<b>Preconditions</b>	Add number has executed successfully.
<b>Post-conditions</b>	The ACTOR has removed a non-seeded number.
<b>Author</b>	Mario Lucassen
<b>Assumptions</b>	The ACTOR knows what a non-seeded number is.

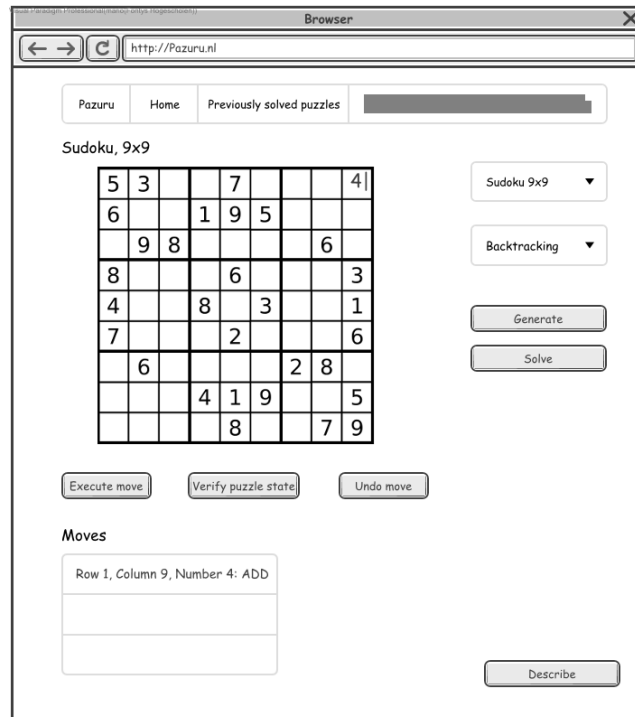
#### 3.2.2.1 Scenarios

##### Happy path

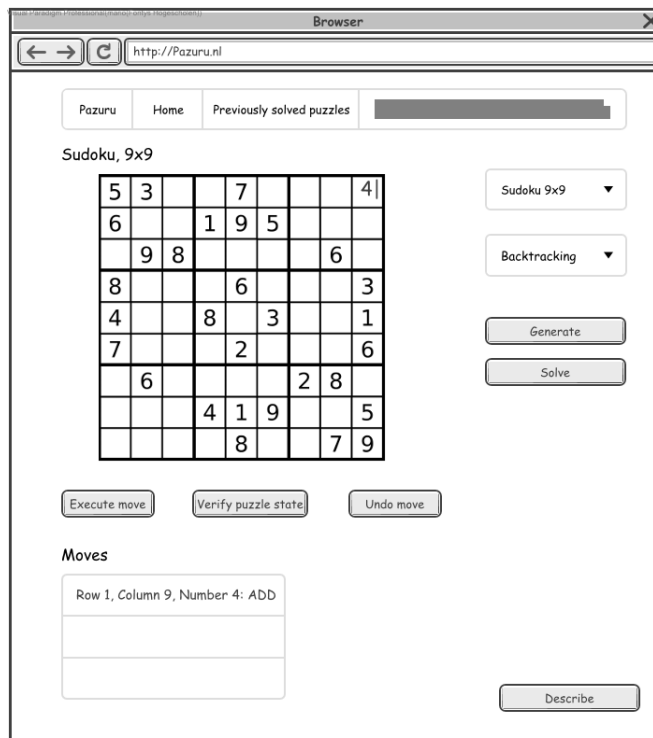
1. The ACTOR clicks on the right most and top most cell in the grid.
2. **SYSTEM** Displays an insertion caret in the designated cell.
3. The ACTOR presses backspace.
4. **SYSTEM** Removes the content from the designated cell
5. The ACTOR clicks on the **Execute move** button.
6. **SYSTEM** Displays and adds the move to the moves list.



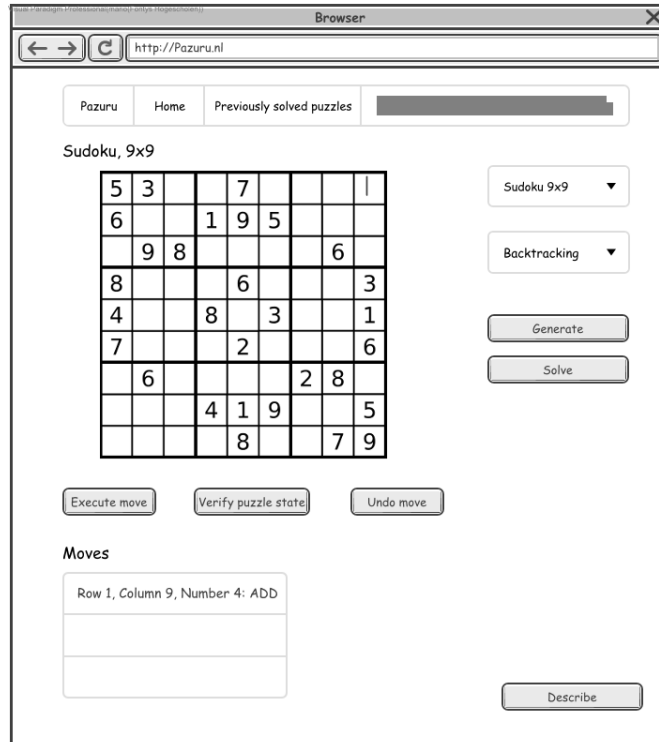
1. The ACTOR clicks on the right most and top most cell in the grid.



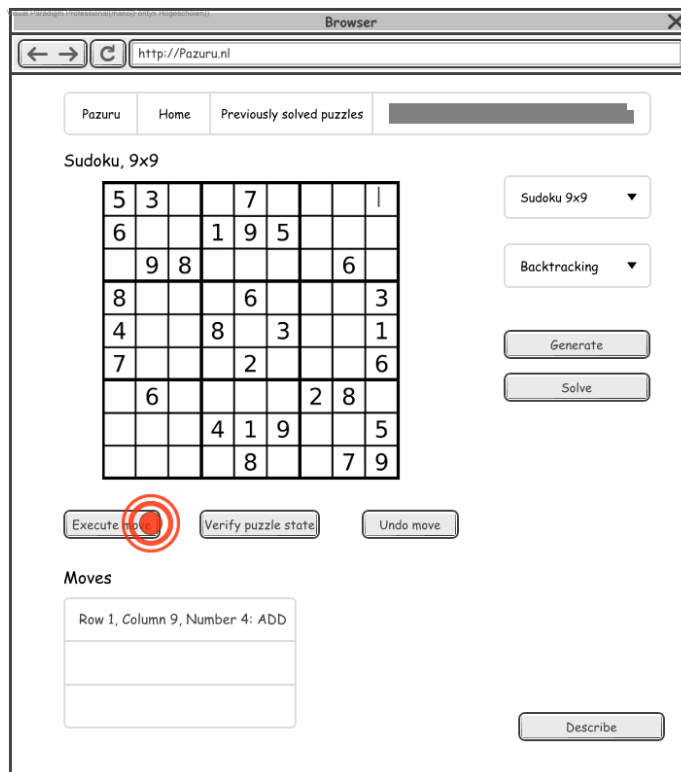
2. SYSTEM Displays an insertion caret in the designated cell.



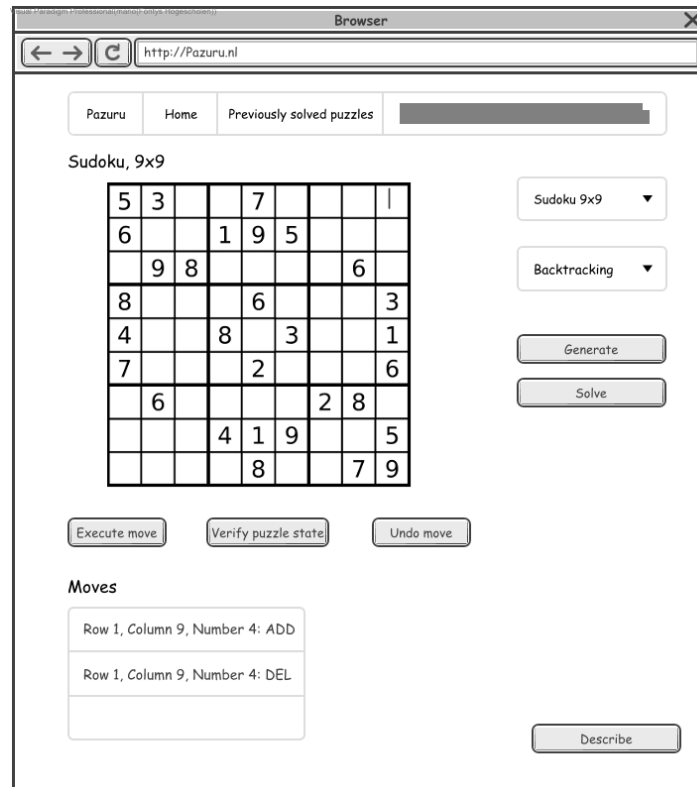
3. The ACTOR presses backspace.



4. SYSTEM Removes the content from the designated cell



5. The ACTOR clicks on the Execute move button.




6. SYSTEM Displays and adds the move to the moves list.



### 3.2.3 Solve Sudoku

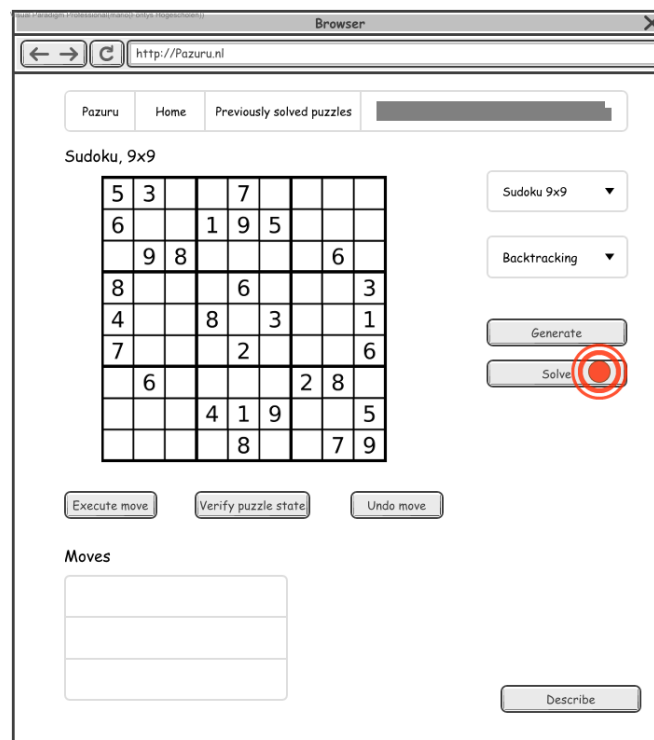
#### Solve Sudoku

<b>ID</b>	UC13
<b>Description</b>	To solve a Sudoku puzzle.
<b>Primary Actors</b>	 Enigmatologist
<b>Level</b>	User
<b>Complexity</b>	Low
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Scheduled
<b>Preconditions</b>	Generate Sudoku has successfully executed with Backtracking selected.
<b>Post-conditions</b>	The Sudoku puzzle grid is now in its solved state.
<b>Author</b>	Mario Lucassen
<b>Assumptions</b>	N/A

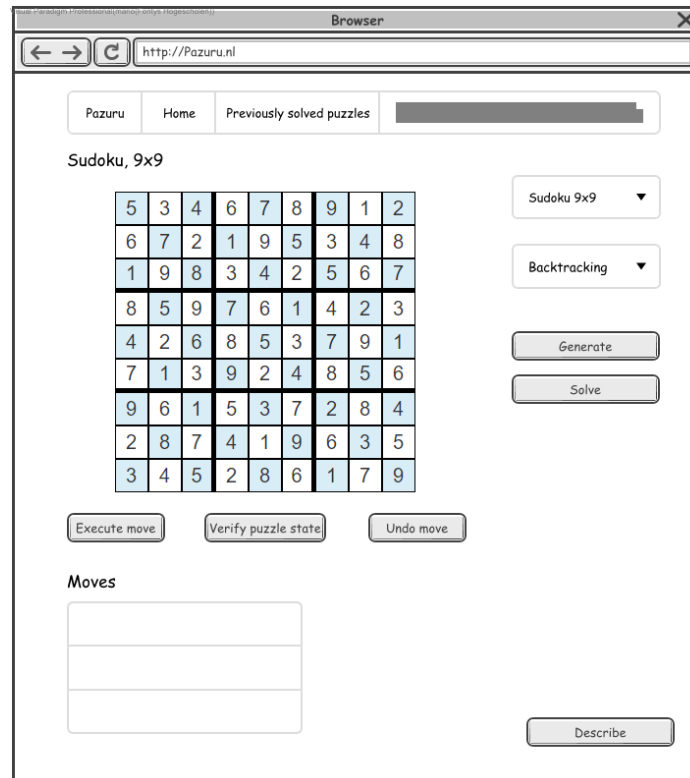
#### 3.2.3.1 Scenarios

##### Happy path

1. The ACTOR clicks on the **Solve** button.
2. **SYSTEM** Displays the Sudoku grid solution.




1. The ACTOR clicks on the Solve button.



2. SYSTEM Displays the Sudoku grid solution.

### 3.2.4 Generate Sudoku

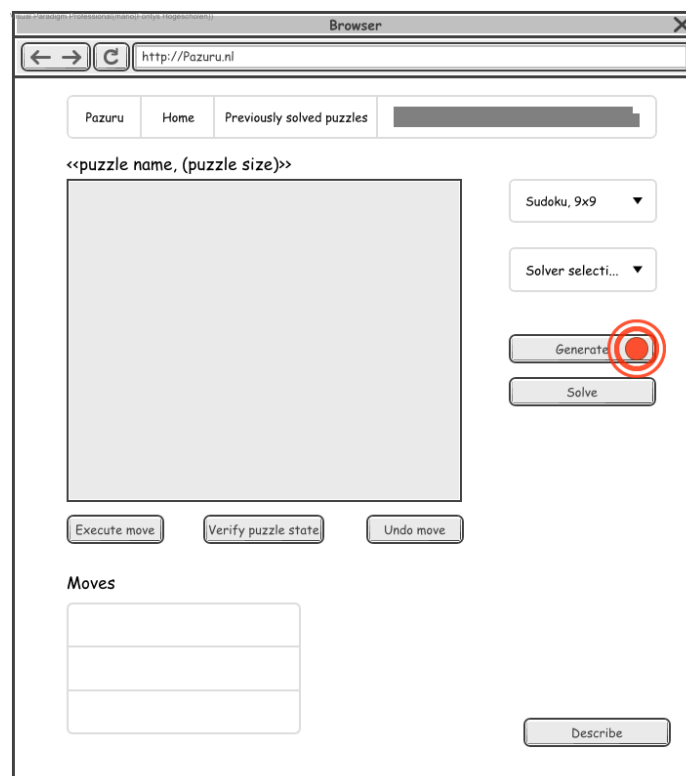
#### Generate Sudoku

<b>ID</b>	UC14
<b>Description</b>	To generate a Sudoku puzzle.
<b>Primary Actors</b>	 Enigmatologist
<b>Level</b>	User
<b>Complexity</b>	Low
<b>Use Case Status</b>	Complete
<b>Implementation Status</b>	Scheduled
<b>Preconditions</b>	Select puzzle has successfully executed with Sudoku as the selected puzzle.
<b>Post-conditions</b>	The ACTOR has generated a Sudoku puzzle.
<b>Author</b>	Mario Lucassen
<b>Assumptions</b>	The Sudoku puzzle is implemented in the system.

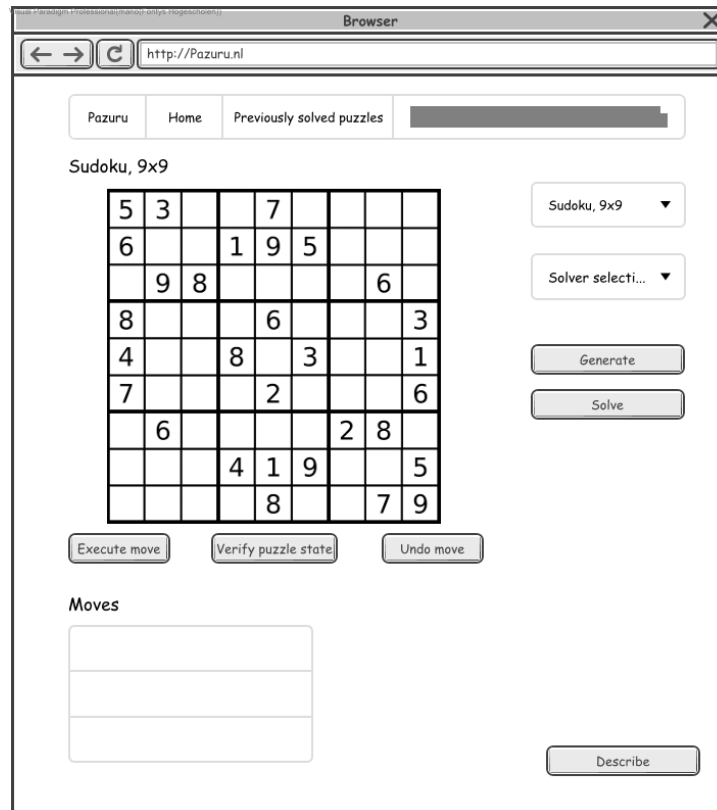
#### 3.2.4.1 Scenarios

##### Happy path

1. The ACTOR clicks on the **Generate** button.
2. **SYSTEM** Displays a Sudoku grid and sets the puzzle name to Sudoku, 9x9.



1. The ACTOR clicks on the Generate button.



2. SYSTEM Displays a Sudoku grid and sets the puzzle name to Sudoku, 9x9.

## 4 User interface specification

This chapter contains the user interface designs. All user interface designs are made with Visual Paradigm wireframe designer.

### 4.1 Home page (Sudoku generated)

The home page contains all Sudoku functionalities as described in chapter 2.1 Actions.

Visual Paradigm Professional (name: only's Hogeschoolen)

Browser

← → ↻ http://Pazuru.nl

Pazuru Home Previously solved puzzles

Sudoku, 9x9

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Sudoku, 9x9 ▼

Solver selecti... ▼

Generate

Solve

Execute move Verify puzzle state Undo move

Moves


Describe

## 5 Solving algorithm

The chosen algorithm for solving a Sudoku puzzle state is a backtracking-algorithm.

### **Pseudocode backtracking-algorithm:**

Find row, col of an unassigned cell

If there is none, return true

For digits from 1 to 9

a) If there is no conflict for digit at row, col

assign digit to row, col and recursively try fill in rest of grid

b) If recursion successful, return true

c) Else, remove digit and try another

If all digits have been tried and nothing worked, return false

The **Big O notation** for the backtracking-algorithm is  $O(n^m)$  where  $n$  stands for the range of possible numbers (for Sudoku that is 1 to 9) and  $m$  stands for the empty cells left over in the puzzle grid. This means that the algorithm is linear and thus a Sudoku puzzle can be solved in linear time.

## 6 Generating algorithm

The chosen algorithm for generating a Sudoku puzzle is a naive-algorithm.

### **Pseudocode naive-algorithm:**

Fill all the diagonal 3x3 sub-grids.

Fill recursively rest of the non-diagonal sub-grids.

For every cell to be filled, we try all numbers until  
we find a safe number to be placed.

Once the grid is fully filled, remove K elements randomly.

In the algorithm **K** stands for the amount of clues to be removed from the puzzle. This can variate between 64 and 4. The **K** will be set to any number in the range from 50 to 63. If **K** is 63 that means 81 minus 63 is 18 and the Sudoku puzzle will have 18 clues (18 seeded cells).

## 7 Puzzle state representation

A classic Sudoku puzzle contains 81 cells. A Sudoku puzzle state can be seen as a String that has a length of 81 characters.

**Sudoku puzzle state as String example:**

"034007008080065000000300070200000700710040096005000001050002000000170060600900430"

The algorithm to depict what number is on **N** row and **M** column where **L** is the puzzle size.

### **Pseudocode getting a number:**

Index is N times L plus M

In string get char at Index

### **Pseudocode setting a number:**

Index is N times L plus M

In string set char at Index