Author information

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

PAZURU SOFTWARE ANALYSIS DOCUMENT

Document no:	1	Created:	09.05.2019
Version:	1.00	Last changes:	10.05.2019
File:	Pazuru Software Analysis Document.docx		

Version history

Version	Date	Changes	Author[1]
0.01	08.05.2019	Document created with Introduction, Requirements and empty Use cases.	Lucassen, Mario M. M.lucassen@student.fontys.nl
0.02	09.05.2019	Updated the Introduction added User Interface	Lucassen, Mario M. M.lucassen@student.fontys.nl
0.03	10.05.2019	Added use case diagram, use cases, interfaces and architectural analysis.	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

[1] Author's name with email address

Table of contents

Table 1. Global definitions.

Name	Description
Enigmatologist	Someone who studies and writes mathematical, word or logic puzzles.
Fontys	Fontys University of Applied Sciences.
Canvas	A portal for students to submit their work to.

1. Introduction

1.1. Purpose

For the course Software Engineering, at the Fontys University of Applied Sciences, in the third semester the assignment Big Idea is to create a distributed application. I have chosen to create a puzzle platform. The platform is intended for Enigmatologists to play around with logic puzzles. A move system for executing and undoing a move will be used to play puzzles, the enigmatologist will be able to verify the current puzzle state to check whether his or her attempt is correct. Not only will the platform support playing puzzles but also generating new ones, solving puzzles and give descriptions for puzzle strategies and rules. All previously solved puzzles will be saved and shown on a previously solved list page that can be filtered by puzzle type. The puzzles that this platform will support are Sudoku and Hitori. For Sudoku, a backtracking algorithm will be used for solving the puzzle. And Hitori will be solved using solving techniques described in the Hitori specification document.

This idea is accepted by the assessor of the course, Kuijpers, Nico N.H.L. in a concept document submitted on Canvas on 14.04.2019.

1.2. Scope

The scope of the product being made is something new and thus not a subsystem, but does include puzzles as subsystems. 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.
Sudoku (subsystem)
Hitori (subsystem)

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 puzzle platform 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 very generic because that is what this platform is. Each puzzle will be documented separately as a subsystem with its own actions and rules section.

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	Select puzzle	To select a puzzle.	Must	High	UC1
F2	Select solver	To select a solver.	Must	High	UC2
F3	Generate puzzle	To generate a puzzle.	Must	High	UC3
F4	Solve puzzle	To solve a puzzle.	Must	High	UC4
F5	Execute move	To execute a move for a puzzle.	Should	Medium	UC5
F6	Undo move	To undo an executed move for a puzzle.	Should	Medium	UC6
F7	Verify puzzle	To verify a puzzle state.	Should	Medium	UC7
F8	Describe puzzle	To describe a puzzle.	Could	Low	UC8
F9	Show solved puzzles	To show the previously solved puzzle list page	Could	Low	UC9
F10	Filter solved puzzles	To filter the previously solved puzzle list by puzzle type.	Could	Low	UC10

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	The puzzle must be playable.	Must	Medium
R2	The puzzle must be solvable.	Must	Must
R3	A puzzle must have at least one rule to be solvable.	Must	Must

2.3. Quality attributes

The quality attributes for the system are shown in the table with categories used by the ISO 25010[2].

ID	Category ISO 25010	Description
Q1	Performance	The maximum latency is 400ms.
Q2	Performance	The maximum resources that the application can use is not more than 2 GB.
Q3	Performance	The minimum puzzles that can be solved concurrently are 10.
Q4	Compatibility	Documentation describing interfaces is available to show how components communicate.
Q5	Usability	The system can be used by anyone within 10 minutes.
Q6	Reliability	The system has a 99% uptime.
Q7	Reliability	The system is at least 23 hours per day available.
Q8	Security	Any data is securely sent to the server and clients.
Q9	Maintainability	The system is built modularly.
Q10	Maintainability	The source code is written according to the Microsoft code conventions[3] for C# and the Java Code Conventions[4].
Q11	Maintainability	Source code is documented.
Q12	Maintainability	At least 80% of the source code is covered by tests.
Q13	Maintainability	Each puzzle added to the platform must have its rules and actions documented in a separate document named after the puzzle.

[2] ISO 25010: https://iso25000.com/index.php/en/iso-25000-standards/iso-25010

[3] C# Microsoft Code Conventions: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions

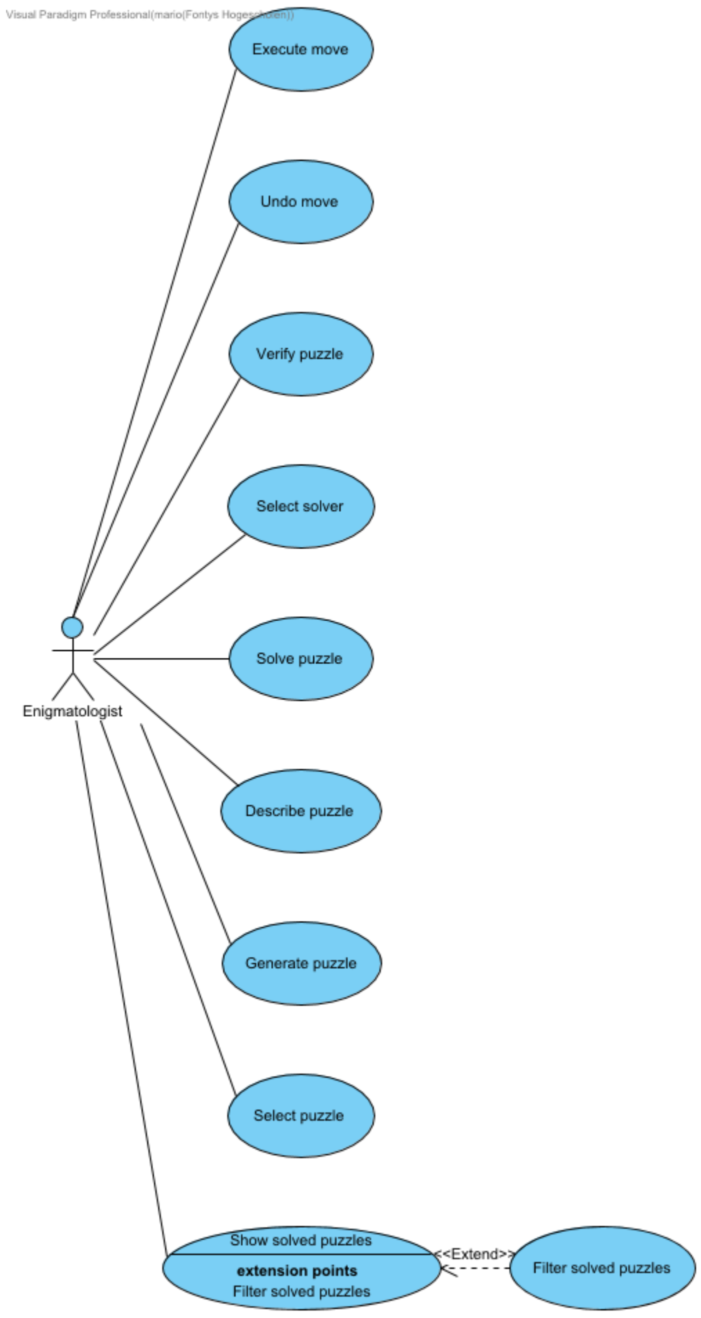
[4] Java Code Conventions: https://www.oracle.com/technetwork/java/codeconventions-150003.pdf

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[5] specification standard.

3.1. Use case diagrams

Pazuru Use case diagram.



[5] Unified Modeling Language Specification 2.5.1: https://www.omg.org/spec/UML/2.5.1

3.2. Use case descriptions

Pazuru Use cases.

3.2.1. Select puzzle

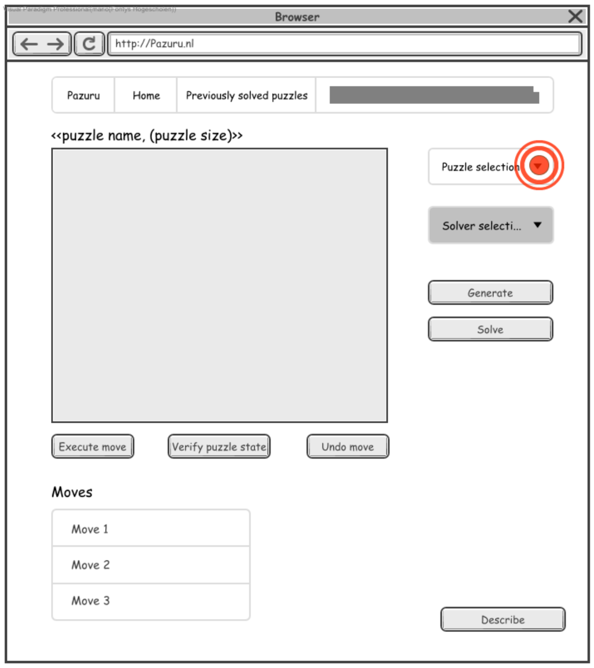
Select Puzzle

ID	UC1
Description	To select a puzzle.
Primary Actors	Enigmatologist
Level	User
Complexity	Low
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	The ACTOR is on the home page.
Post-conditions	The ACTOR has selected the first puzzle in the puzzle selection.
Author	Lucassen, Mario M.
Assumptions	At least one puzzle is implemented in the system.

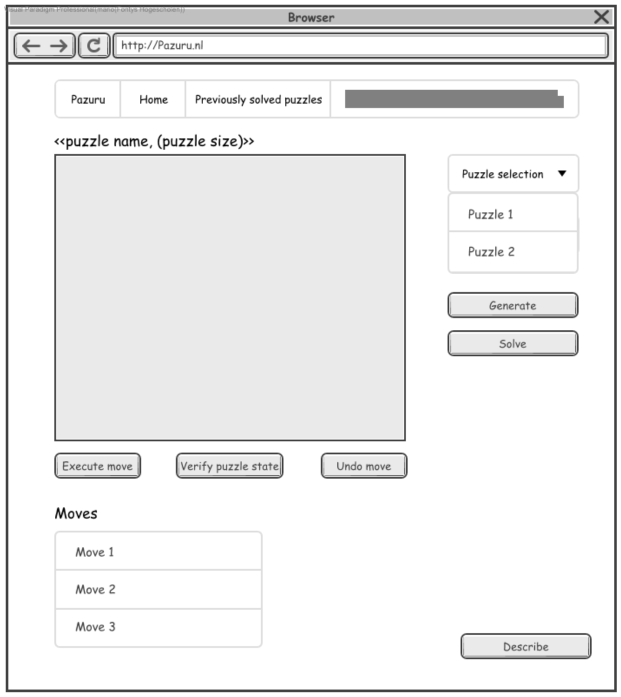
Scenarios

Happy path

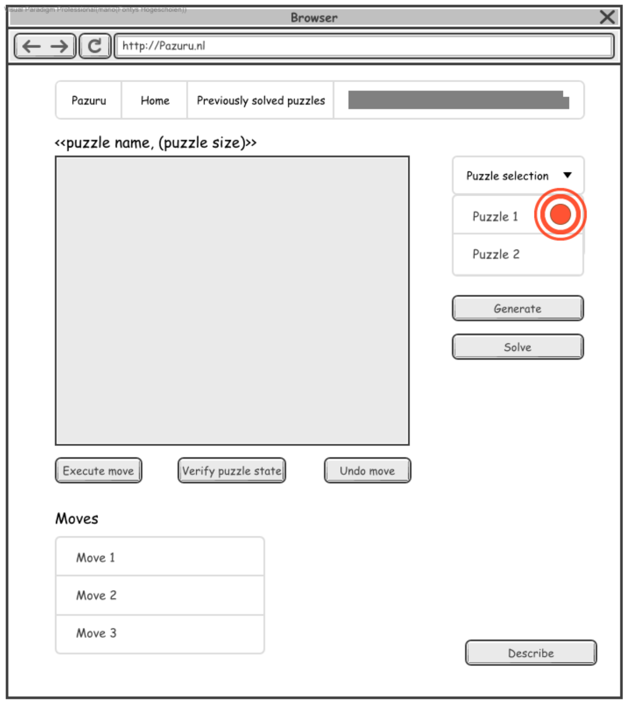
1. The ACTOR clicks on the puzzle select.
2. SYSTEM Displays a list of puzzles.
3. The ACTOR clicks on the first puzzle.
4. SYSTEM Displays the selected puzzle and enables the puzzle solver select.



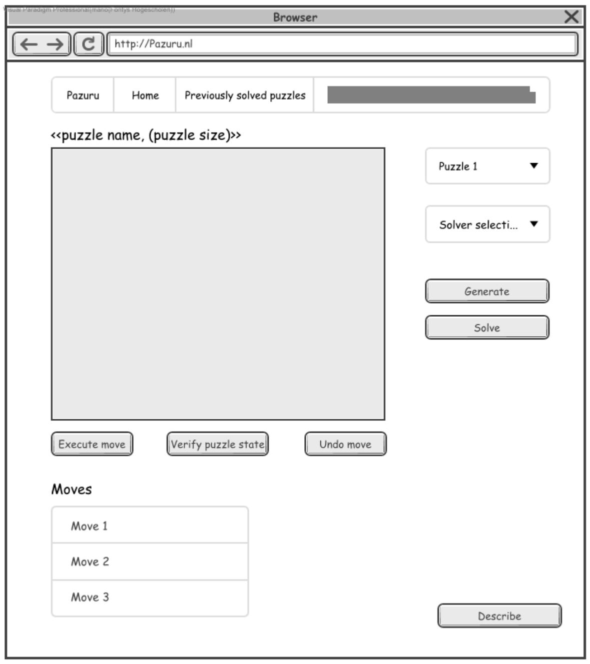
1. The ACTOR clicks on the puzzle select.



2. SYSTEM Displays a list of puzzles.



3. The ACTOR clicks on the first puzzle.



4. SYSTEM Displays the selected puzzle and enables the puzzle solver select.

3.2.2. Select solver

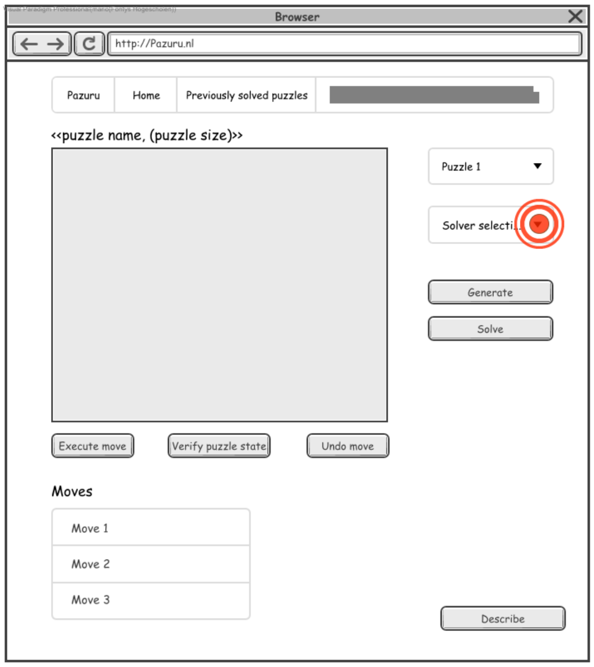
Select solver

ID	UC2
Description	To select a solver.
Primary Actors	Enigmatologist
Level	User
Complexity	Low
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	Select puzzle is successfully executed.
Post-conditions	The ACTOR has selected a solver.
Author	Lucassen, Mario M.
Assumptions	At least one solver is implemented in the system for the selected puzzle.

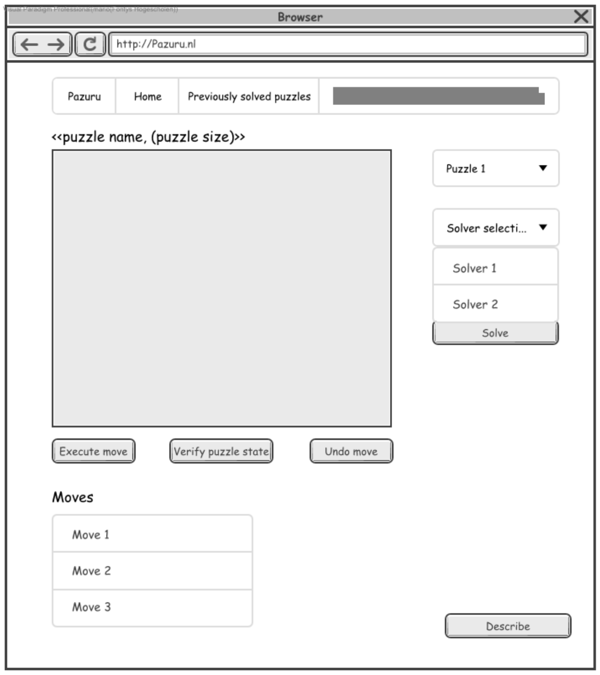
Scenarios

Happy path

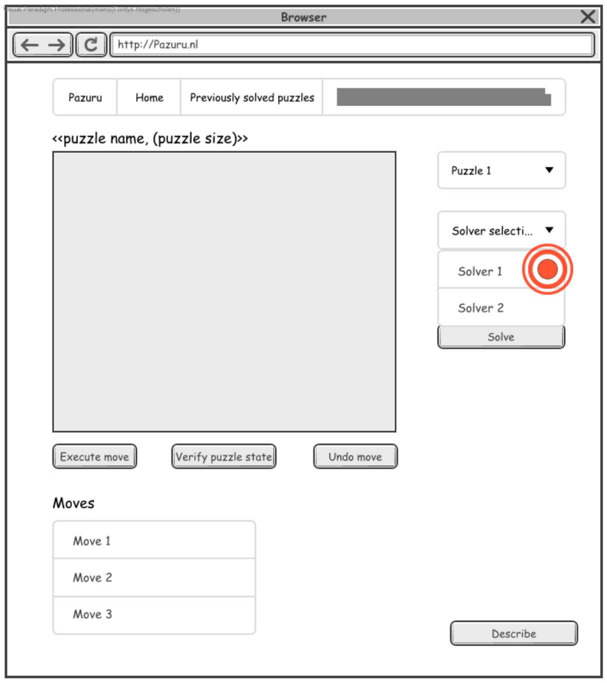
1. The ACTOR clicks on the solver select.
2. SYSTEM Displays a list of solvers for the selected puzzle.
3. The ACTOR clicks on the first solver.
4. SYSTEM Displays the selected solver.



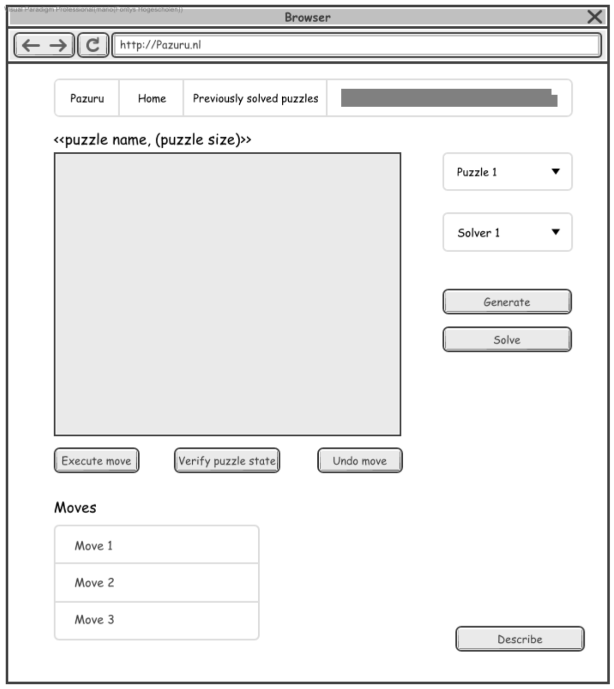
1. The ACTOR clicks on the solver select.



2. SYSTEM Displays a list of solvers for the selected puzzle.



3. The ACTOR clicks on the first solver.



4. SYSTEM Displays the selected solver.

3.2.3. Generate puzzle

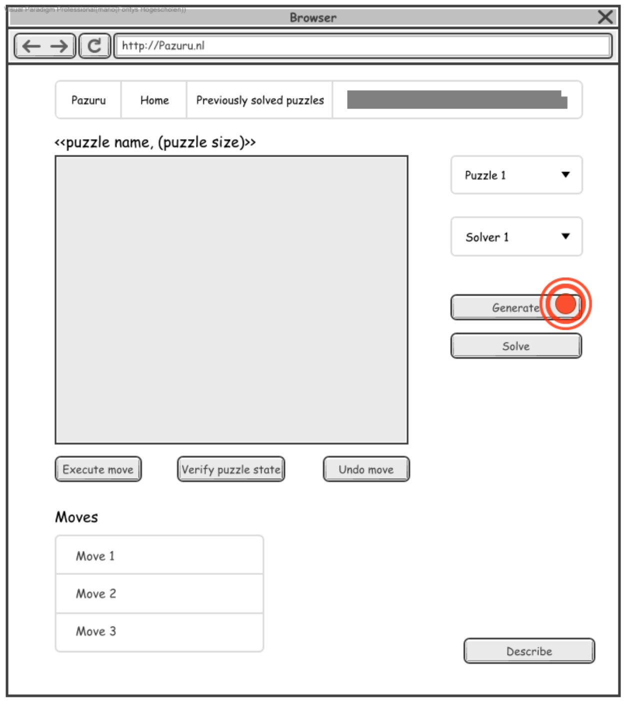
Generate puzzle

ID	UC3
Description	To generate a puzzle.
Primary Actors	Enigmatologist
Level	User
Complexity	Low
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	Select solver is executed successfully.
Post-conditions	The ACTOR has generated a puzzle.
Author	Mario Lucassen
Assumptions	N/A

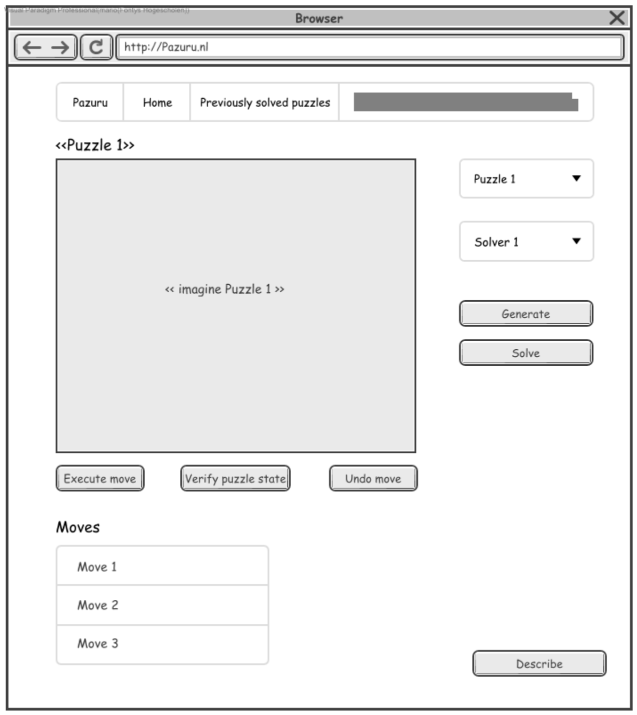
Scenarios

Happy path

1. The ACTOR clicks on the **Generate** button.
2. SYSTEM Displays the generated puzzle and sets the puzzle name.



1. The ACTOR clicks on the Generate button.



2. SYSTEM Displays the generated puzzle and sets the puzzle name.

3.2.4. Solve puzzle

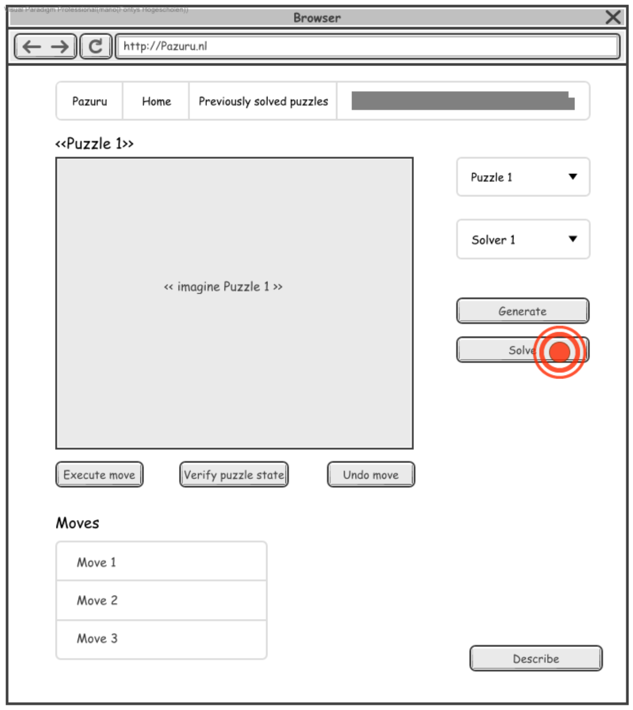
Solve puzzle

ID	UC4
Description	To solve a puzzle.
Primary Actors	Enigmatologist
Level	User
Complexity	Low
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	Generate puzzle is executed successfully.
Post-conditions	The current puzzle is solved.
Author	Mario Lucassen
Assumptions	N/A

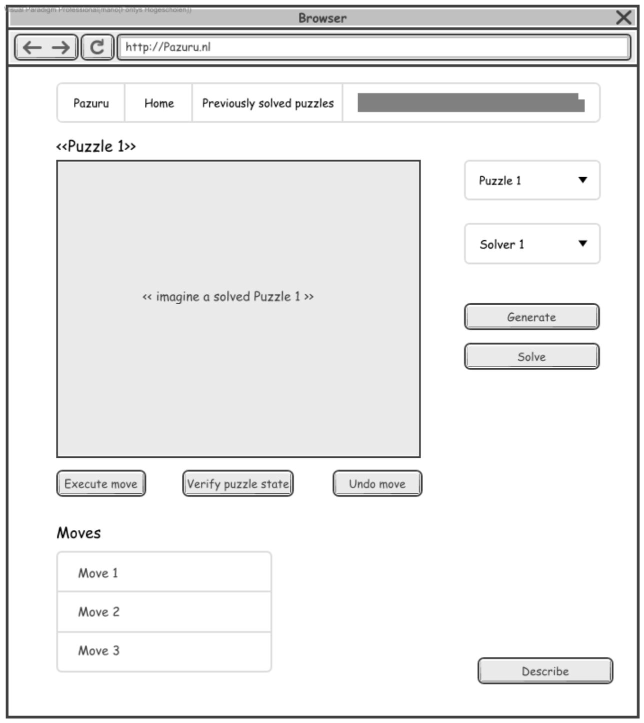
Scenarios

Happy path

1. The ACTOR clicks on the **Solve** button.
2. SYSTEM Displays the solved puzzle solution.



1. The ACTOR clicks on the Solve button.



2. SYSTEM Displays the solved puzzle solution.

3.2.5. Execute move

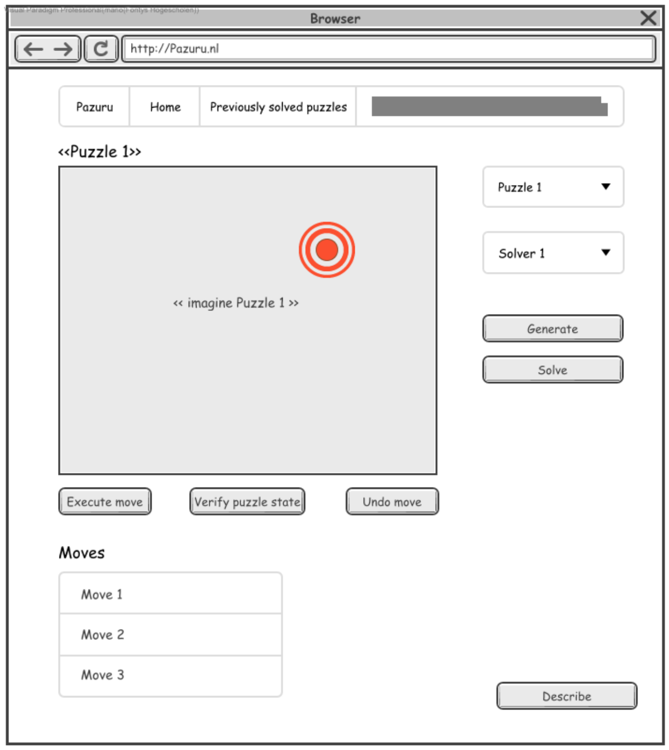
Execute move

ID	UC5
Description	To execute a move for a puzzle.
Primary Actors	Enigmatologist
Level	User
Complexity	Medium
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	Generate puzzle is executed successfully.
Post-conditions	The ACTOR has successfully executed a move.
Author	Mario Lucassen
Assumptions	The ACTOR knows how to perform a move for the selected puzzle.

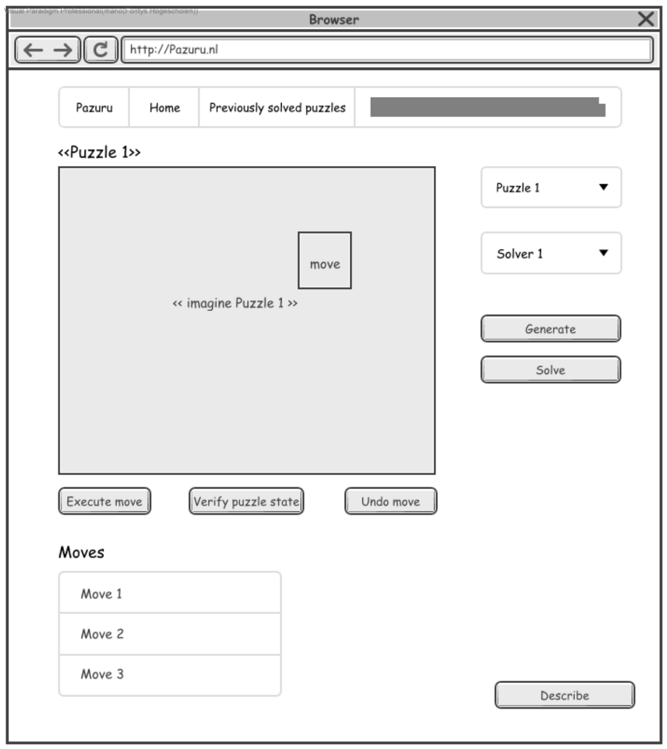
Scenarios

Happy path

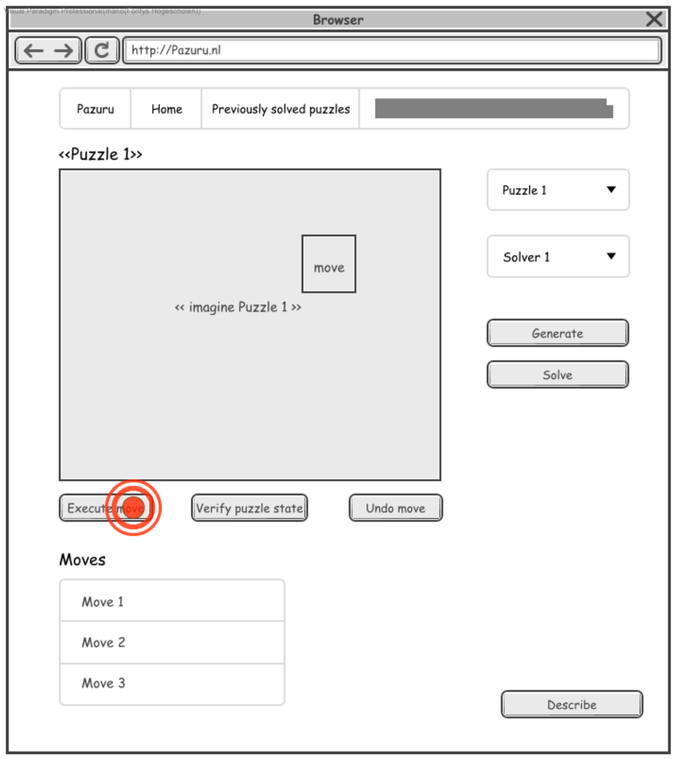
1. The ACTOR clicks on the puzzle to select a move.
2. SYSTEM Displays that a move is selected.
3. The ACTOR clicks on the **Execute move** button.
4. SYSTEM Performs the executed move, adds the move to the moves list and displays the new puzzle state.



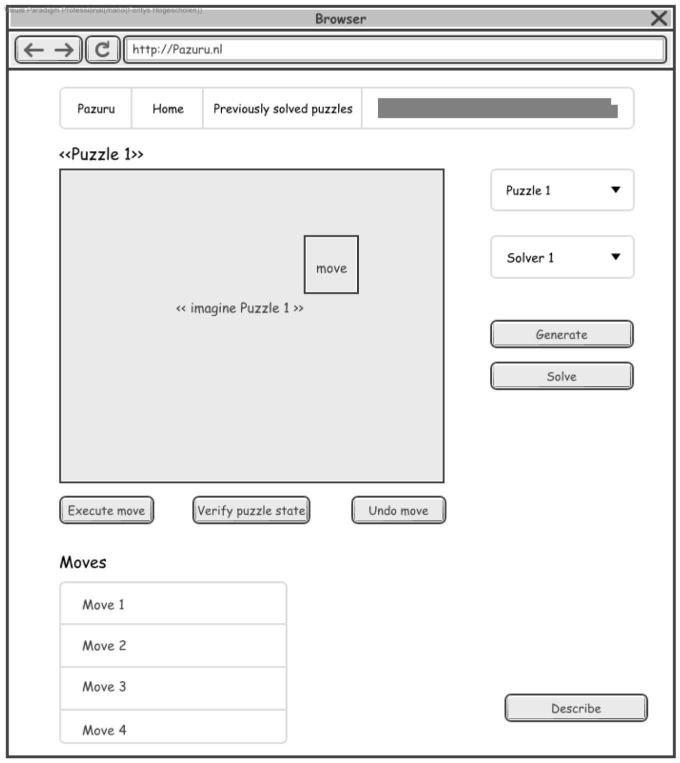
1. The ACTOR clicks on the puzzle to select a move.



2. SYSTEM Displays that a move is selected.



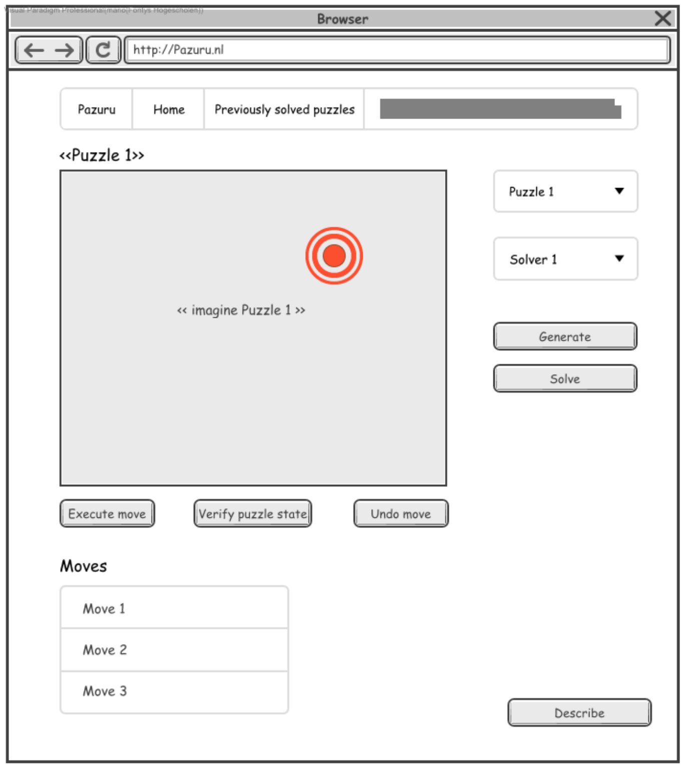
3. The ACTOR clicks on the Execute move button.



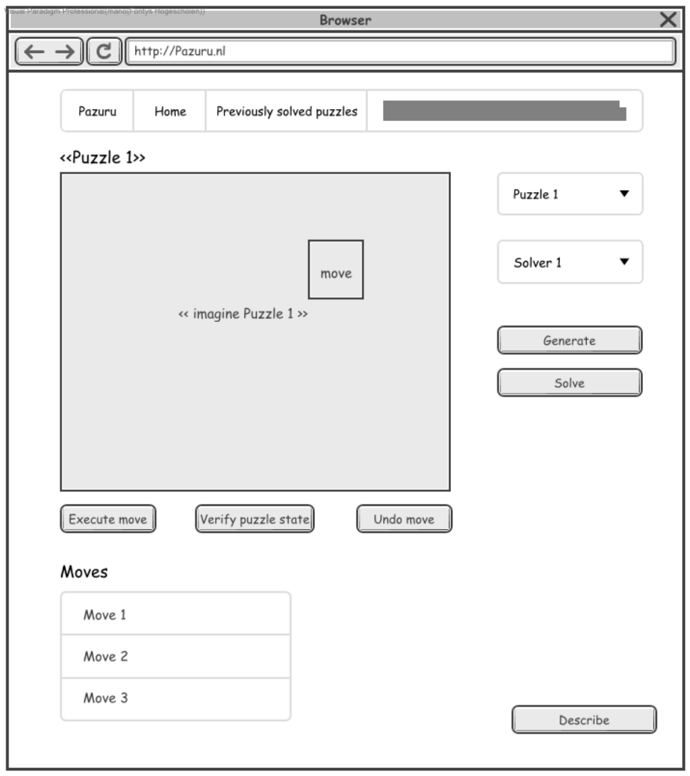
4. SYSTEM Performs the executed move, adds the move to the moves list and displays the new puzzle state.

Invalid move path

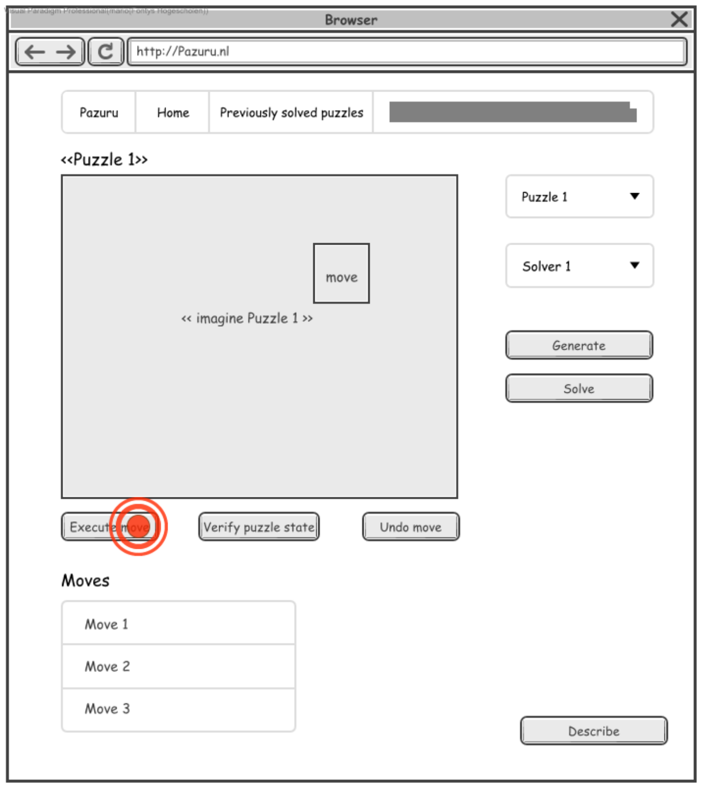
1. The ACTOR clicks on the puzzle to select a move.
2. SYSTEM Displays that a move is selected.
3. The ACTOR clicks on the **Execute move** button.
4. SYSTEM Displays popup message that the move is invalid.
5. The ACTOR clicks on **Ok** button.
6. SYSTEM Removes the popup message and removes the invalid move.



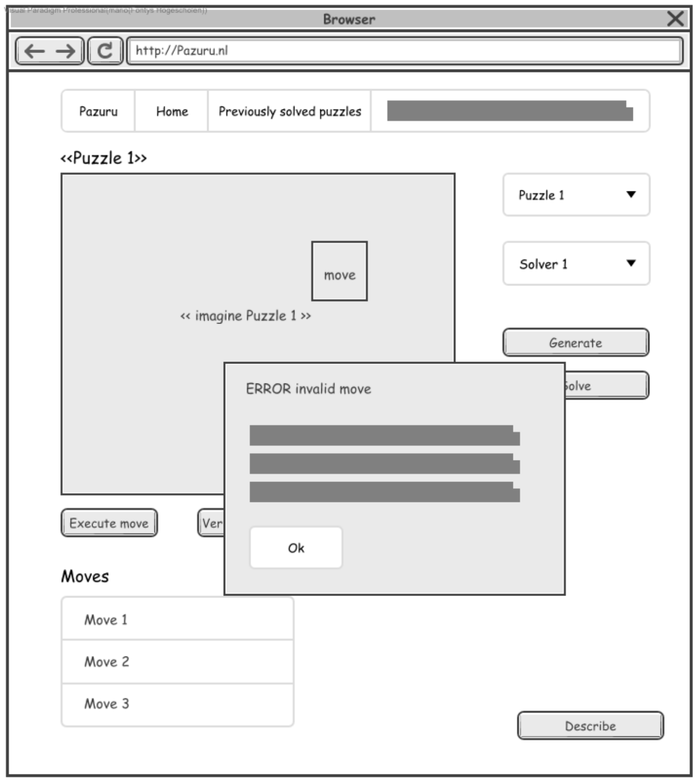
1. The ACTOR clicks on the puzzle to select a move.



2. SYSTEM Displays that a move is selected.

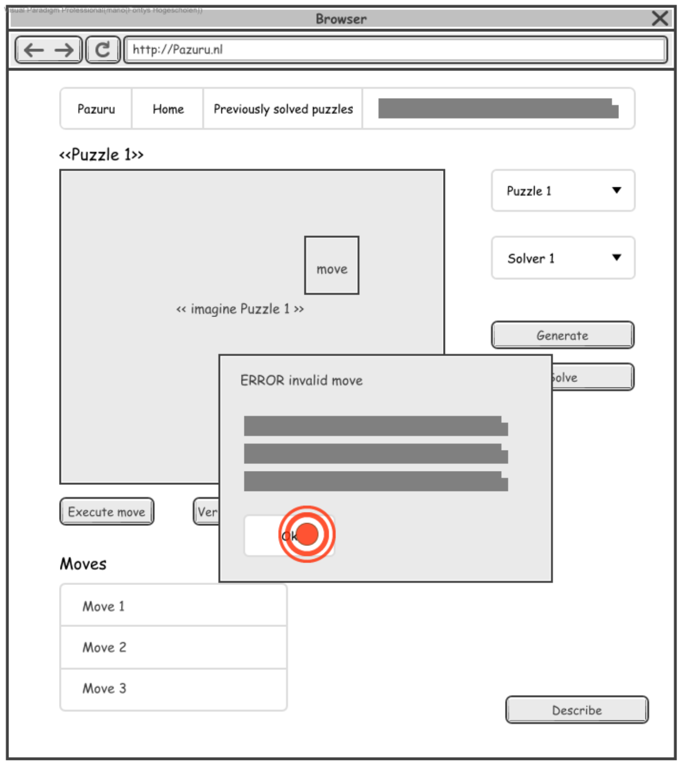


3. The ACTOR clicks on the Execute move button.



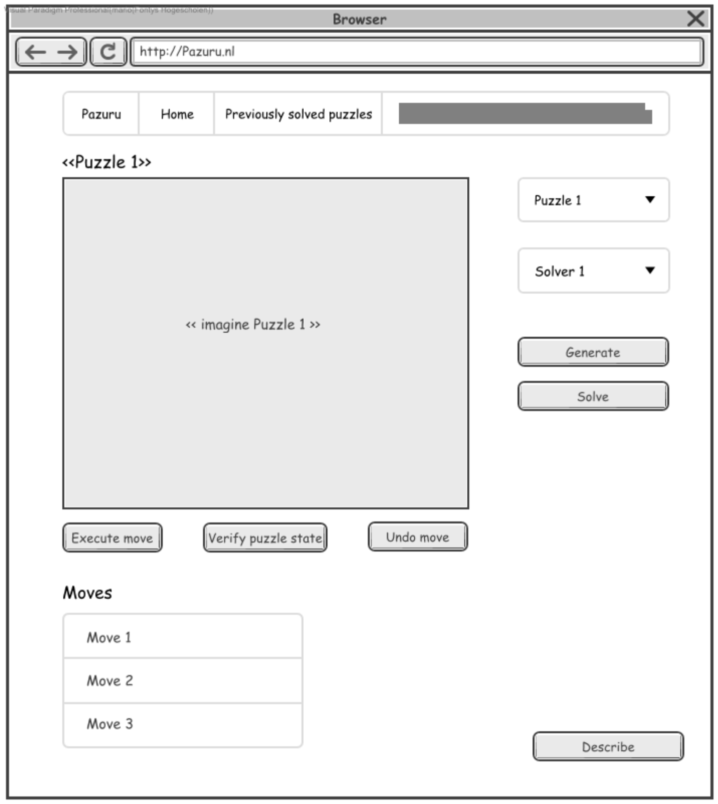
ERROR invalid move

4. SYSTEM Displays popup message that the move is invalid.



ERROR invalid move

5. The ACTOR clicks on Ok button.



6. SYSTEM Removes the popup message and removes the invalid move.

3.2.6. Undo move

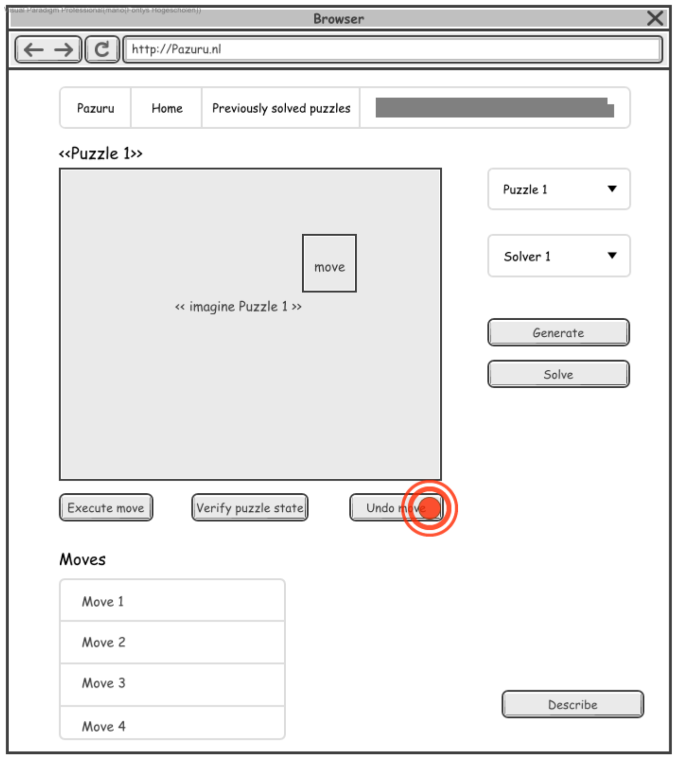
Undo move

ID	UC6
Description	To undo an executed move for a puzzle.
Primary Actors	Enigmatologist
Level	User
Complexity	Low
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	Execute move is executed successfully.
Post-conditions	The ACTOR has successfully executed an undo move.
Author	Mario Lucassen
Assumptions	N/A

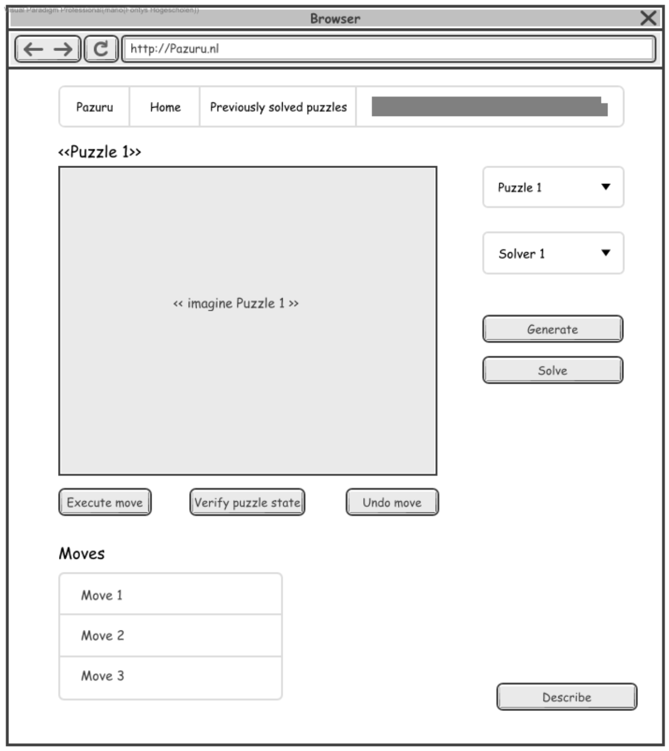
Scenarios

Happy path

1. The ACTOR clicks on the **Undo move** button.
2. SYSTEM Performs the undo move, removes the move from the moves list and displays the previous puzzle state.



1. The ACTOR clicks on the Undo move button.



2. SYSTEM Performs the undo move, removes the move from the moves list and displays the previous puzzle state.

3.2.7. Verify puzzle

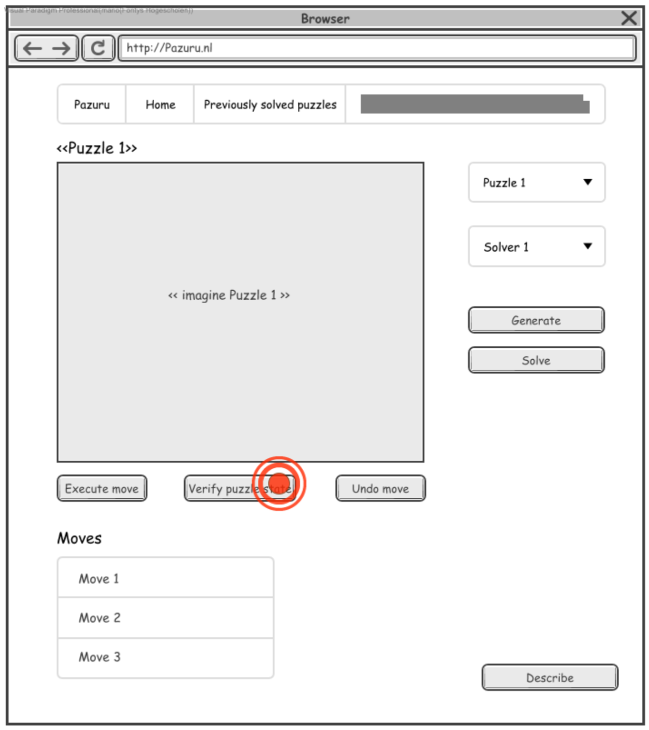
Verify puzzle

ID	UC7
Description	To verify a puzzle state.
Primary Actors	Enigmatologist
Level	User
Complexity	Low
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	Generate puzzle is executed successfully.
Post-conditions	The puzzle state is verified.
Author	Mario Lucassen
Assumptions	N/A

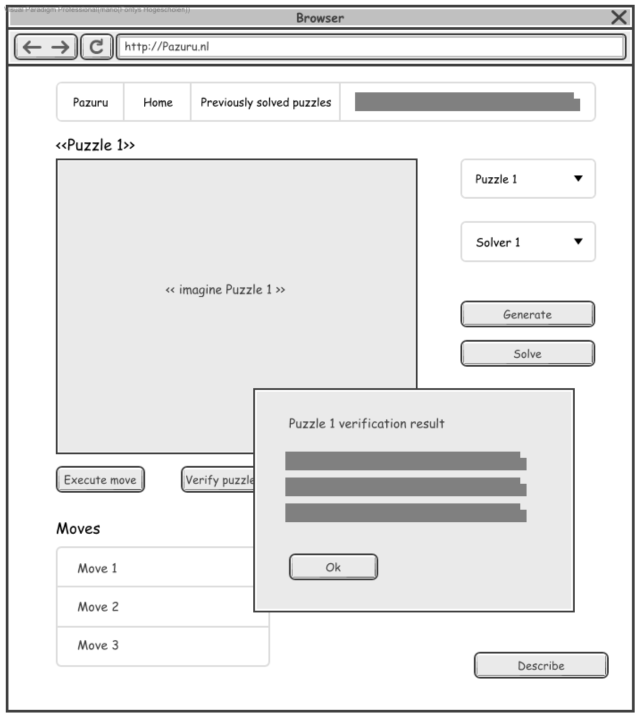
Scenarios

Happy path

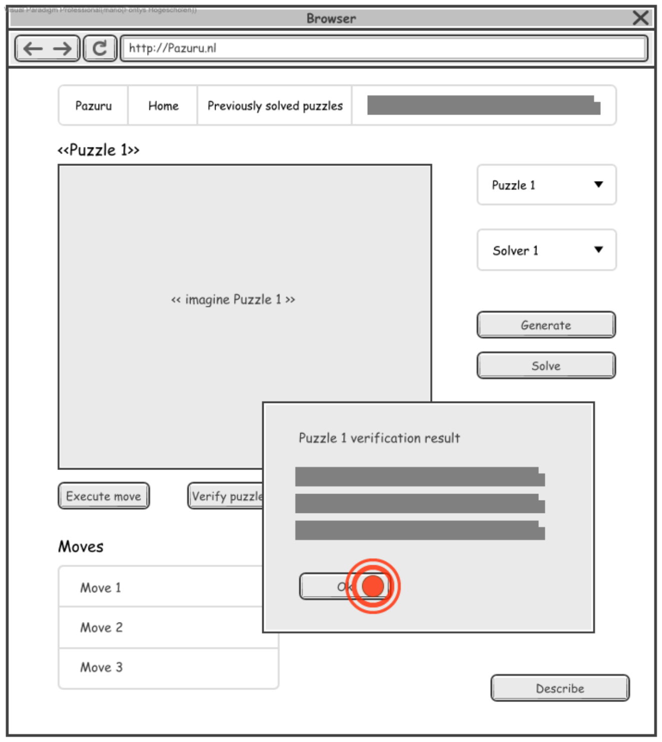
1. The ACTOR clicks on the **Verify puzzle state** button.
2. SYSTEM Displays a popup with the verification result of the puzzle state.
3. The ACTOR clicks on the **Ok** button.
4. SYSTEM Removes the popup.



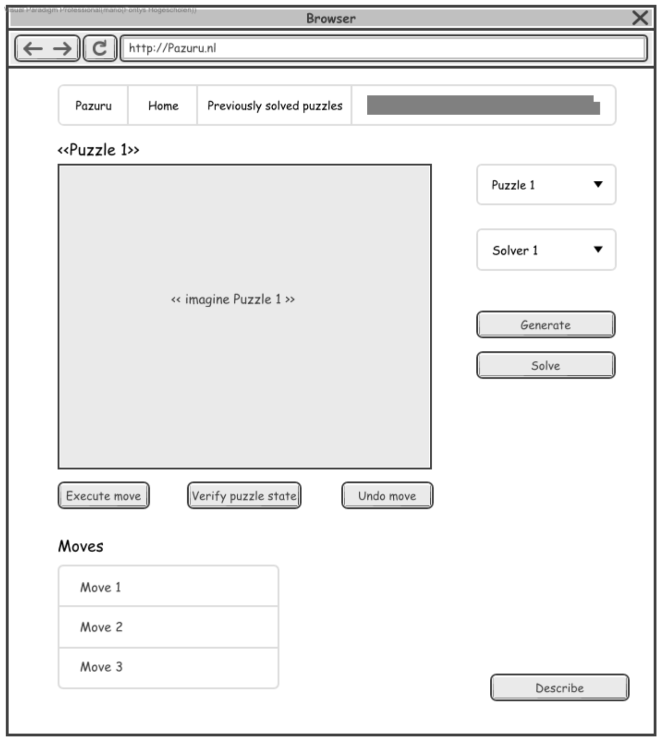
1. The ACTOR clicks on the Verify puzzle state button.



2. SYSTEM Displays a popup with the verification result of the puzzle state.



3. The ACTOR clicks on the Ok button.



4. SYSTEM Removes the popup.

3.2.8. Describe puzzle

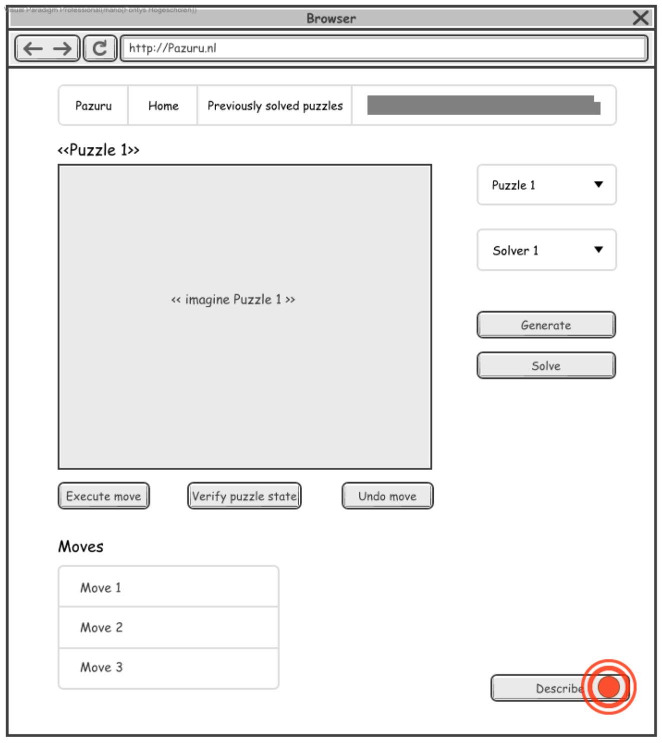
Describe puzzle

ID	UC8
Description	To describe a puzzle.
Primary Actors	Enigmatologist
Level	User
Complexity	Low
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	The ACTOR is on the Home page.
Post-conditions	The ACTOR has the popup that describes the puzzle open.
Author	Mario Lucassen
Assumptions	N/A

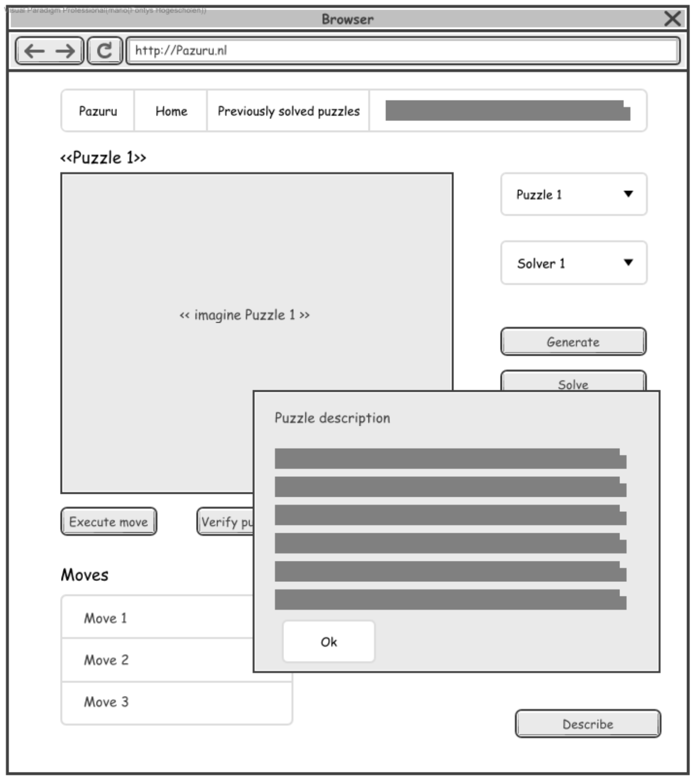
Scenarios

Happy path

1. The ACTOR clicks on the **Describe** button.
2. SYSTEM Displays a popup that describes the puzzle.



1. The ACTOR clicks on the Describe button.



2. SYSTEM Displays a popup that describes the puzzle.

3.2.9. Show solved puzzles

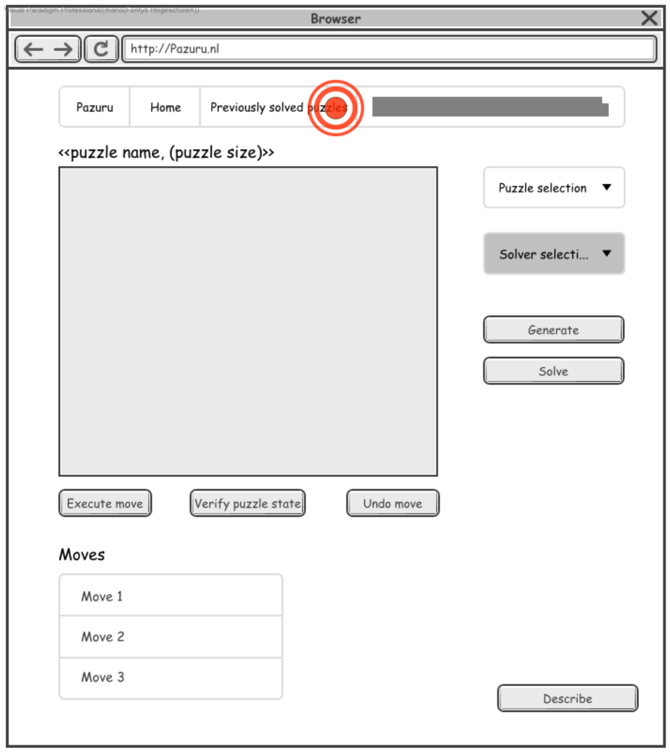
Show solved puzzles

ID	UC9
Description	To show the previously solved puzzle list page.
Primary Actors	Enigmatologist
Level	User
Complexity	Low
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	The ACTOR is on the Home page.
Post-conditions	The ACTOR has seen the solved puzzles page.
Author	Mario Lucassen
Assumptions	N/A

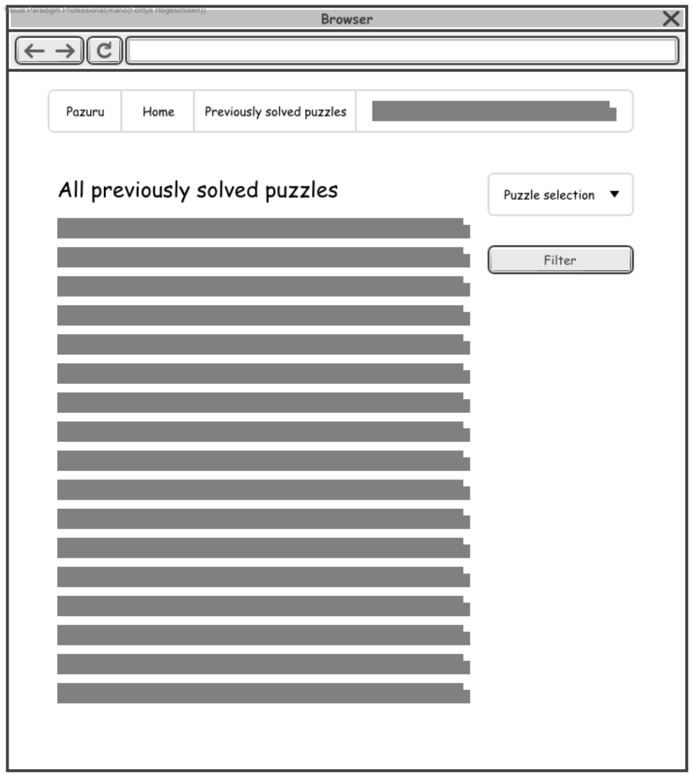
Scenarios

Happy path

1. The ACTOR clicks on the **Previously solved puzzles** tab.
2. SYSTEM Displays the **Previously solved puzzles** page.



1. The ACTOR clicks on the Previously solved puzzles tab.



2. SYSTEM Displays the Previously solved puzzles page.

3.2.10. Filter solved puzzles

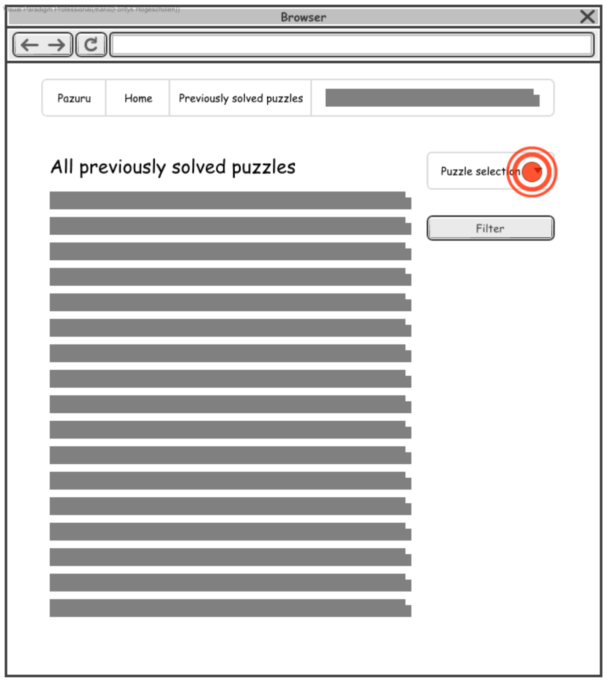
Filter solved puzzles

ID	UC10
Description	To filter the previously solved puzzle list by puzzle type.
Primary Actors	Enigmatologist
Level	User
Complexity	Low
Use Case Status	Complete
Implementation Status	Scheduled
Preconditions	The ACTOR is on the Previously solved puzzles page.
Post-conditions	The ACTOR has filtered the solved puzzles list.
Author	Mario Lucassen
Assumptions	N/A

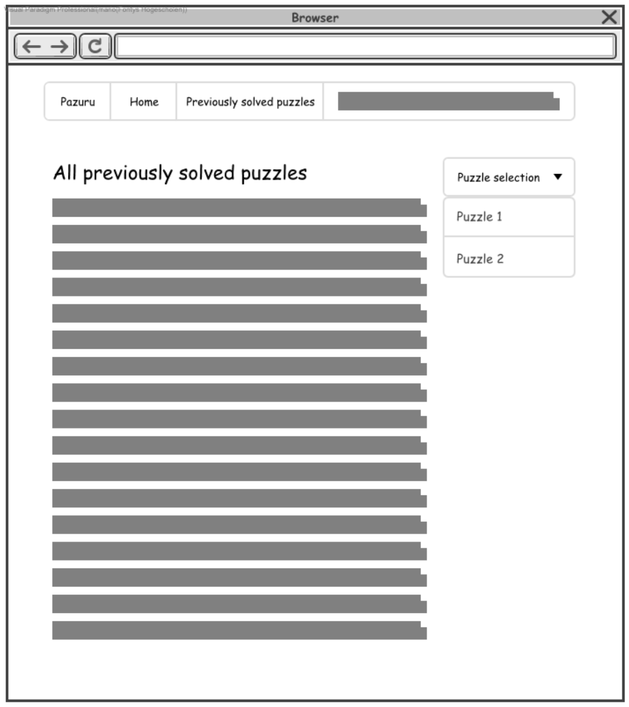
Scenarios

Happy path

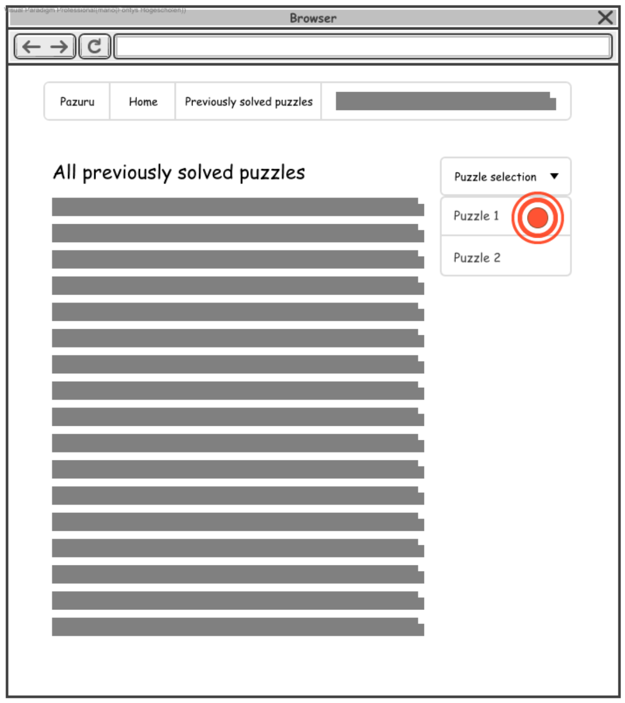
1. The ACTOR clicks on the puzzle selection.
2. SYSTEM Displays puzzle type list.
3. The ACTOR clicks on the first puzzle.
4. SYSTEM Displays the selected puzzle in the puzzle selection.
5. The ACTOR clicks on the **Filter** button.
6. SYSTEM Filters and only displays solved puzzles from the selected type.



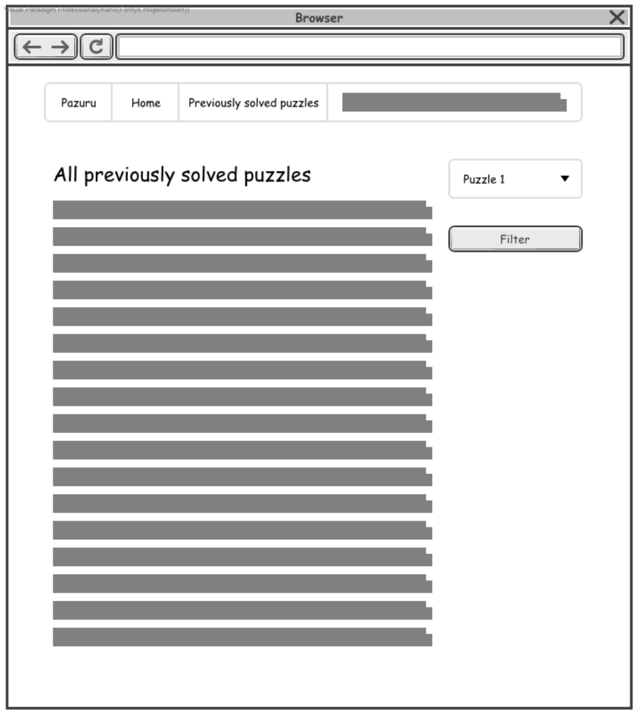
1. The ACTOR clicks on the puzzle selection.



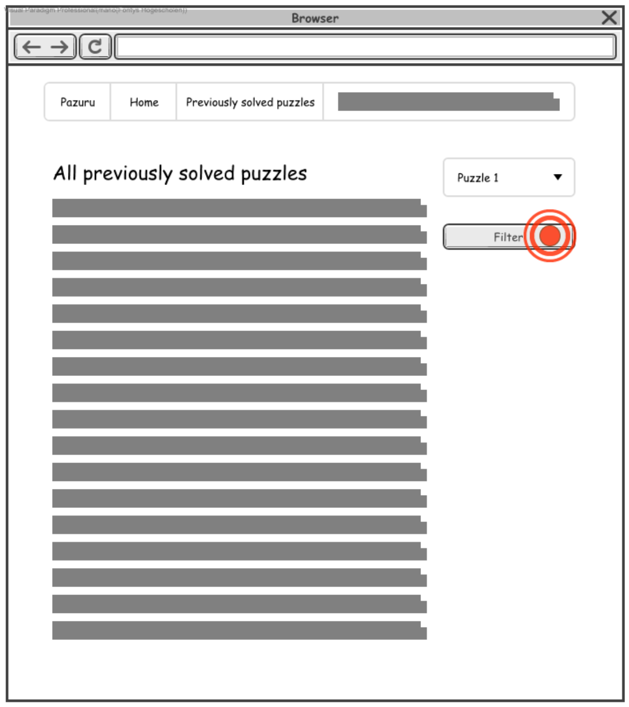
2. SYSTEM Displays puzzle type list.



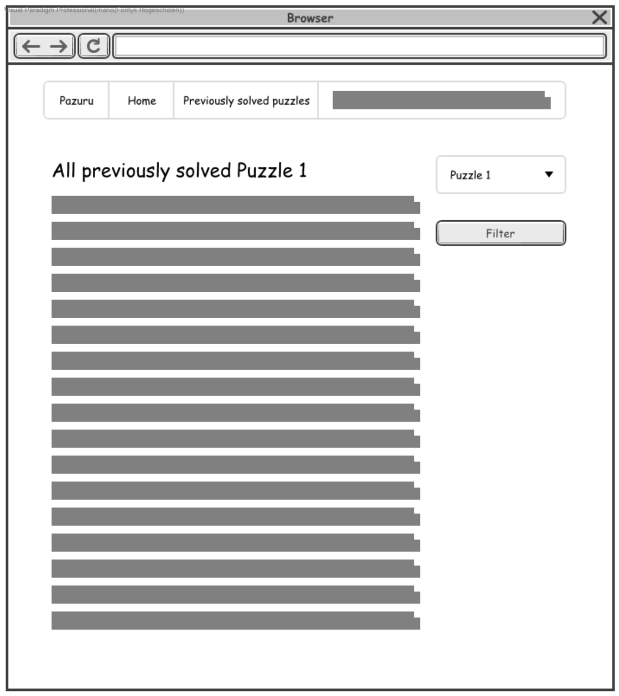
3. The ACTOR clicks on the first puzzle.



4. SYSTEM Displays the selected puzzle in the puzzle selection.



5. The ACTOR clicks on the Filter button.



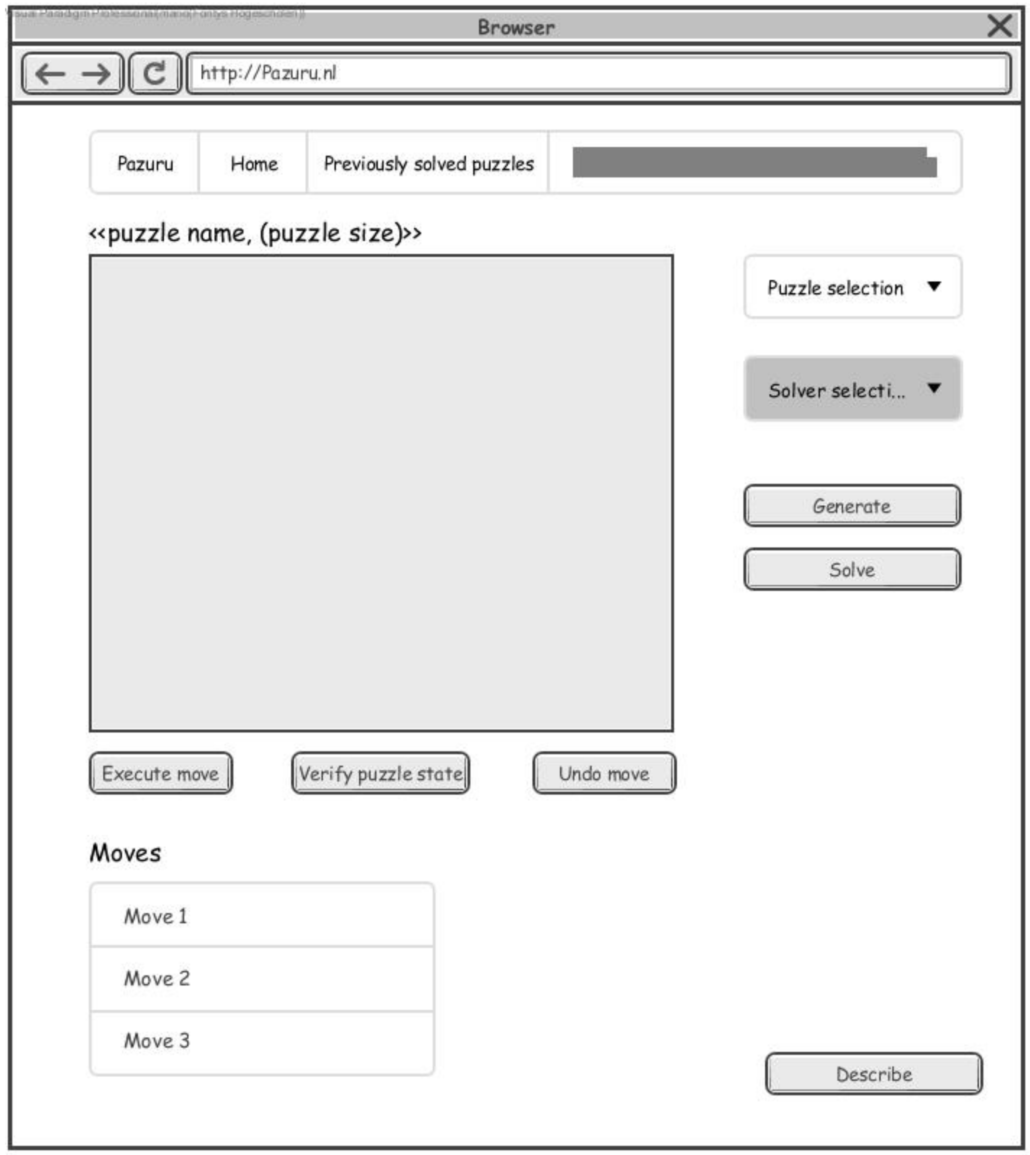
6. SYSTEM Filters and only displays solved puzzles from the selected type.

4. User interface specification

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

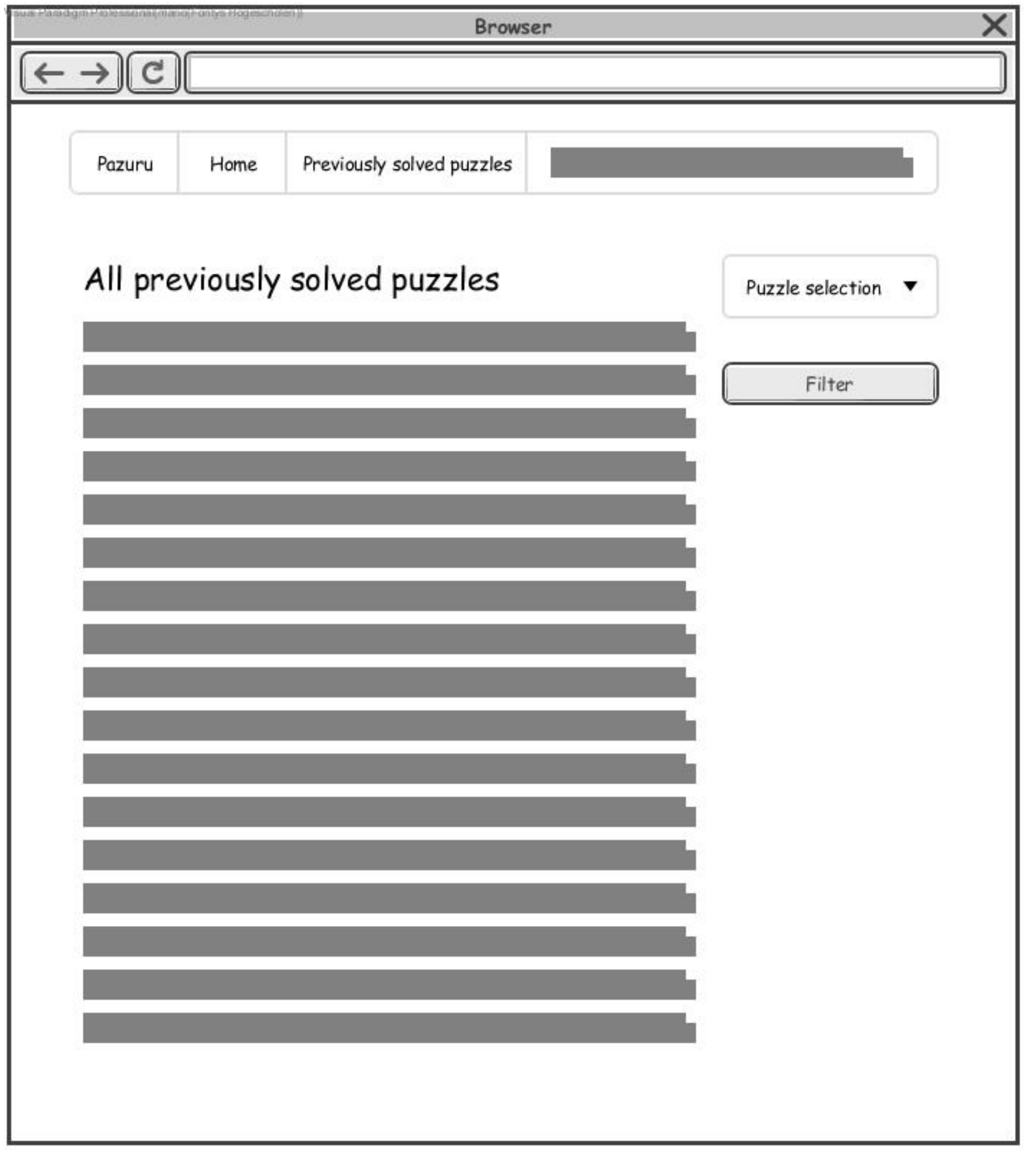
4.1. Home page

The home page contains the main functionalities as described in chapter 2.1 Actions, F1 to F8.



4.2. Previously solved puzzles page

The previously solved puzzles page contains the functionalities as described in chapter 2.1 Actions, F9 to F10.


Visual Paradigm Professional(mario(Fontys Hogescholen))
Browser
Pazuru
Home
Previously solved puzzles
All previously solved puzzles
Puzzle selection
Filter

5. Architectural analysis

This chapter will give a glance of the system that will be built. This will also be documented in the Software Architecture document but more elaborate.

5.1. System overview

The overall structure of the system being built is shown in the table below.

Nodes
Pazuru Client
Pazuru Server
Pazuru REST API

5.1.1.Client

The client will handle user inputs and send them to the server and receive responses from the server.

5.1.2.Server

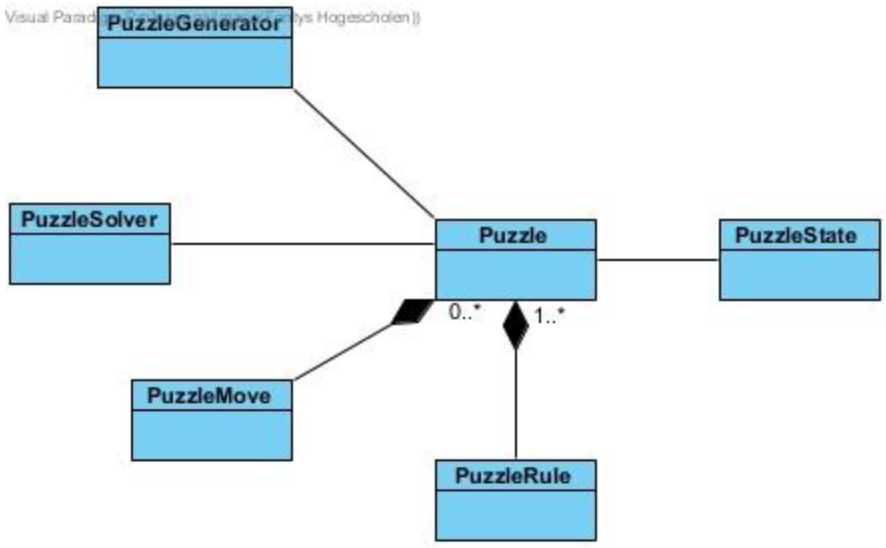
The server will handle all logic for the puzzles, such as executing a puzzle move or generating a new puzzle. The server can communicate with the Client but also with the REST API to store generated and solved puzzles.

5.1.3.REST API

The REST API its primary use is to store generated and solved puzzles in a database.

5.2. Analysis object model

The Domain model for the system. These classes are derived from the use cases.



Explanation

A puzzle has 1 or more rules, as required by the requirement rules in chapter 2.2 Rules. A puzzle also has a collection of moves for the Move system to execute and undo. A puzzle has 1 PuzzleState to describe the current puzzle state.