Requirements Document

Visualisation of Relations in Mathematics

Michal Jaroš
Jakub Fraňo
Filip Kotoč
Eva Ivanusyková

Contents

1 Introduction

1.1 Purpose of requirements document

This document serves as a system requirements specification document for the Information Systems Development project for the academic year 2024/25. The document serves as a legally binding agreement between the project client and the development team over the project's scope and aims to clearly outline the client's needs that were discussed in joint meetings. The document is primarily intended for the client, the development team, and other individuals involved in the management and development of this project.

1.2 Scope of the product

This application is designed to assist mathematicians by providing graphical representations of the connections between axiomatic subsets and the interdependencies among statements. The final application will feature an interactive graph that allows users to explore and search for relationships between mathematical statements.

1.3 Definitions, acronyms and abbreviations

- **Graph** - a structure consisting of a set of objects where some pairs of the objects are in some sense "related". The objects are represented by abstractions called nodes (also called nodes or points) and each of the related pairs of nodes is called an edge.
- **Theory** - a set of statements.
- **Canvas** - place where a graph is visualised.
- **Theorem** - a formula, proposition, or statement in mathematics or logic deduced or to be deduced from other formulas or propositions. *In this document we will be using word Statement instead of Theorem / Proof of Theorem.*
- **Axioms** - in logic, axioms are an indemonstrable first principle, rule, or maxim, that has found general acceptance or is thought worthy of common acceptance whether by virtue of a claim to intrinsic merit or on the basis of an appeal to self-evidence.
- **Node** - the fundamental unit of which graphs are formed x
- **Metamath** - simple and flexible computer-processable language that supports rigorously verifying, archiving, and presenting mathematical proofs. Metamath has a small specification that enables you to state a collection of axioms (assumptions), statements, and proofs (aka a "database").

1.4 References

- GitHub repository of the project
- Project Metamath
- Project Metamath overview

1.5 Overview of the remainder of the document

In the following chapters, the reader will learn about the various functionalities, detailed requirements and will get an idea of how user can work with the application. The information system's general description is covered in the second chapter. A thorough explanation of each feature is covered in the third chapter.

2 General description

2.1 Product perspective

The purpose of the application is to assist mathematicians by graphically visualizing mathematical connections as a graph. User can create his own theory which is visualized as a graph by adding statements which are visualised as nodes of the graph. After that user can choose between 2 main graph types. The first one named Statements depending on the theory 3.3.1 shows all statements that prove a user defined theory. The second one named Dependencies of the theory 3.3.2 shows all statements contributed by the defined theory. These representations help mathematicians understand and visualise relations between mathematical statements.

2.2 Product functions

The process of using the application begins with user uploading (for example drag and drop) a database in .mm format. Once the database is uploaded, the user can eather import a previously saved theory or utilize the search bar to find specific statements and create a theory. The user can either proceed further or create another theory to work with in the next steps.

Next, the user chooses the type of visualisation they wish to generate. There are two main visualisation options. The first options named Statements depending on the theory 3.3.1 hierarchically displays a graph of statements, that are needed to prove each theory. The second options named Dependencies of the theory 3.3.2 similarly displays a graph of statements that can be proven using each theory. If the user created or loaded 2 theories, the graphs of both theories intersect highlighting their common statements.

This visual representation can be exported both as an image (bitmap format) and as a graph structure for further use (can be imported as mentioned before). On the side of the screen can be seen information about created theory, such as number of statements or axioms. Additionally, when the user hovers over any node in the graph, they will receive detailed information (for example number of axioms, theorems and definitions needed for proving this node) about the statement it represents.

Overall, the application provides a dynamic and interactive way to visualize mathematical theories and their relationships, offering flexibility in depth and representation while ensuring users can export and analyze the data efficiently.

2.3 User characteristics

There is only one type of user, that has access to all features of the application.

2.4 General constraints and dependencies

The project is a web applications. Users will need a web browser, internet connection and metamath database in order to access the application. This application assumes that dependencies of statements does not contain cycles.

3 Specific requirements

3.1 Application startup

(a) User opens application by opening index.html file on his local server.

(b) When the application opens, user will see white screen with two buttons and a text in the middle.

(c) Text is name of an application, short description about what the application does and short instruction on how to import the file and start the application.

(d) First button is for selecting a file in metamath (.mm) file format. Second button is for submitting the file.

(e) If the user selects a file in wrong format, with incorrect syntax or did not select any file, the error message will be shown on the screen describing the problem. After the error message the user can again select a file and try again.

(f) User can only click on the submit button, if database is selected.

(g) If the user clicks on the submit button, the user interface will be shown to the user.

3.2 Search

(a) User has a search bar available with a button on the side.

(b) The button switches between two modes: searching through the database to add a statement to a theory and searching through an already drawn graph 3.3 to highlight a statement.

(c) The default is "Add statements" mode which means that the search bar is used for adding statements to theories 3.2.1. User can also choose "Search for nodes" 3.2.2 mode.

3.2.1 Add statements

(a) User has two theories available. Theories are sets of statements which user can manipulate by adding statements (j) to the theory or deleting statements (l) from the theory.

(b) Theories are named as Theory 1 and Theory 2.

(c) If the theory is selected, it means that after adding a selected statement (j) the statement will be added to the theory that is selected.

(d) By default, theories are empty, and Theory 1 is selected.

(e) At all times, exactly one theory is selected. If user selects un-selected theory, the other one will be un-selected.

(f) User can search for the statements using the search bar by entering either the name of the statement (as it is represented in .mm file) or word/phrase from the description of the statement in the .mm file.

(g) The application will give search suggestions ordered by the match under the search bar.

(h) Suggestion has a name of the statement and short description of the statement.

(i) Maximum number of suggestions is 5.

(j) After user finds the desired statement, he can click it and it will be added to the selected theory.

(k) Theory cannot contain a statement twice.

(l) Statement can be deleted from the theory by double clicking the statement.

3.2.2 Search for nodes

(a) This type of search is available only when the graph is visualized.

(b) The user can search for statements already visualized in the graph.

(c) The application will give search suggestions ordered by the match under the search bar.

(d) Suggestion has a name of the statement and short description of the statement.

(e) Maximum number of suggestions is 5.

(f) If the graph does not contain the statement, no statement is found, otherwise the desired statement is highlighted.

3.3 Graph visualisation

(a) Before visualising the graph user can choose from two graph visualisation options. One is named Statements depending on the theory 3.3.1 and the other is named Dependencies of the theory 3.3.2.

(b) **Statements depending on the theory** is selected by default.

(c) User can switch to the other visualisation option by clicking the "Switch type" button. The name of the visualisation option will be placed somewhere on the screen so the user knows which one is active.

(d) If the user has at least one statement in at least one theory, he can click on the button named "Visualise" that is next to the "Switch type" button.

(e) After clicking on the "Visualise" button the hierarchically ordered graph will be shown, it is made out of nodes and edges where nodes represent the statements and edges represent connections between nodes.

(f) Nodes representing the axioms, definitions, statements in theories, hypotheses, and other statements are visualised differently from each other allowing user to easily distinct between them.

(g) The graph is hierarchically ordered meaning that each node is at some set depth from the theories, which is visualised on the graph by putting the nodes that are on the same depth on the same "level" on the screen.

(h) User has 2 option to choose from to manage how graph will look based on depth. First option sets depth of node as maximal possible, meaning if there are 2 possible paths to that node the program will choose the longer one. Second option sets node to lowest depth from his paths.

(i) The layout of nodes in one layer will be optimized for viewing.

(j) The graph is transitively reduced, meaning that when node c implies b, which implies c, and when c implies a, there will not be an edge from c to a, because they are already connected through the b.

(k) The graph will show the statements from the selected theories on the same level. If the **Statements depending on the theory** is chosen all the other nodes will be above the statements from theories. If the Dependencies of the theory is chosen all the other nodes will be below the statements from theories.

Lets assume for now that only one theory has statements in it. Behavior with two theories having statements is described in 3.3.3

(l) Depending on what type of graph visualisation is chosen, different graph will be shown.

(m) User can choose the depth of graph, in default it will be set to infinity.

3.3.1 Statements depending on the theory

(n) In this graph will be visualised all statements that were **supported by theory statements**. That means that chosen theory (by user in the beginning) and no other statements was needed in order to prove all these displayed statements.

3.3.2 Dependencies of the theory

(o) In this graph will be visualised all statements that are **supporting theory statements**. That means, that all these statements were needed for proving chosen theory (by user in the beginning).

3.3.3 Two theories

(p) Behavior with two theories not empty is same as in **Dependencies of the theory** and **Statements depending on the theory** but at all times the nodes that are connected to theory 1 and theory 2 will be highlighted with color and grouped together. (for example theory 1 would be in color red, theory 2 in color blue and intersection in color purple)

3.4 Manipulation with graph

(a) The user can move around the pane where the graph is visualized in all four directions using the mouse.

(b) The user can also zoom in and out using the scroll wheel.

(c) The user can move nodes on the graph by dragging and dropping them on the pane.

(d) The user has the ability to click on a node to select it.

(e) If the node is selected, information about the statement/the proof that it represents will be shown on the screen. The information is: Full name of the statement/the proof, its description, its distance from the theory, the whole proof, distinct variable groups, allowed substitution hints, axioms that it is proven from, definitions that this statement depends on and statements that are referenced by it.

(f) While having some node selected, when user hovers with mouse over other nodes the name of the node that is hovered over the name of the statement and distance from the selected node will be shown.

(g) While hovering over a node and not having any node selected, the name of the node will be shown.

(h) On the screen there's information visible about the whole graph. Specifically number of all nodes, number of axioms and definitions.

(i) Aplication have also few useful checkboxes for better manipulation with graph:

  (1) if we are using Statements depending on the theory:
    (a) axioms must also be specified in the theory
    (b) cannot use other statements - all statements from the theory must be used
    (c) can use other statements (that aren't includet in created theory) - Other starting points than those are allowed, in theory
  (2) if we are using Dependencies of the theory:
    (a) show axioms
    (b) include only nodes that are used in the proof of each statement in the theory
  (3) for both above scenarios:
    (a) show only edges between adjacent floors/show all edges

3.5 Import/export

(a) User can see Import/Export section in which there's a drop down menu, and two buttons.

(b) In drop down menu, user can choose what type of export he wants, the options are:

(1) Image - exports and image of the graph in png format.

(2) Graph - user will get a graph in some standard graph file format.

(3) Statements - user will get a list of all the statements that he has in his theories in some appropriate file format. This file is meant for the future import.

(c) After the user chooses what option he wants, he is able to click on Export button next to the drop down menu. This will download the file.

(d) User can import a theory by clicking the appropriately marked button.

(e) If the uploaded file is not in a required file format (it should be the file that is exported in (3)), or the file is not in a correct syntax, the error message will be shown describing the problem.

(f) If the file is correct the user's theories will get filled with the statements from the file.

Návrh

Visualisation of Relations in Mathematics

Michal Jaroš, Jakub Fraňo, Filip Kotoč, Eva Ivanusykova

1. Úvod

Účel dokumentu

Tento dokument vznikol v rámci predmetu Tvorba informačných systémov v školskom roku 2023/2024 a slúži ako kompletný a detailný návrh aplikácie pre vizualizáciu matematických vzťahov. Obsahuje všetky informácie potrebné pre vysvetlenie a pochopenie funkcionality ako aj spôsobu implementácie systému. Tento dokument je primárne určený pre vývojárov. Obsah v tomto dokumente zahŕňa všetky požiadavky z katalógu požiadaviek

Rozsah využitia systému

Pre prácu s týmto dokumentom je potrebné sa najprv oboznámiť s katalógom požiadaviek. Tento dokument špecifikuje všetky požiadavky z katalógu požiadaviek. Obsahuje celkový návrh používateľského prostredia vrátane vizualizácie. Nachádzajú sa tu aj diagramy, ktoré slúžia na bližší popis implementácie systému a celkový plán implementovania aplikácie.

2. Použité technológie

V tomto projekte budú použité tieto technológie:

- **Cytoscape.js** - vizualizácia grafu
- **metamath-py** - práca s databázou, metódu parse(fpath)
- **Flask -** backend server na python
- **Bootstrap** - ui
- **HTML/CSS** - vizuálna časť
- **JavaScript** - logika aplikácie

Backend

Python server je vytvorený pomocou knižnice **Flask**, tá nám aj určuje celú štruktúru projektu - v priečinku templates sa nachádzajú html súbory, v static máme všetky javascript súbory ktoré pracujú na frontende a v static/style sa nachádzajú css súbory. V súbore _init__.py sa nachádzaju routovacie pravidlá a v súbore routes.py sa nachádza samotný backendový kod. Server sa spúšťa pomocou run.py.

Vytvorenie backendu bolo za účelom zjednodušenia si práce keďže časť potrebného parsera je implementovaná v knižnici **metamath-py**. Z tejto knižnice používame class Statement do ktorej sme pridali atribúty comment, is_referenced_by a proved_from_statements, dalej používame class Database ktorá uchováva zoznam zparsovaních statementov - do tejto triedy sme pridali funkciu fill_references ktorá slúži na naplnenie pridaních atribútov v class Statement. Na samotné parsovanie používame funkciu parse ktorú sme modifikovali aby parsovala aj comment. Na pracovanie s touto knižnicou sme si vytvorili class DataHandler ktorý slúži na pracovanie s databázou čiže má implementované metódy ako napríklad parse_database, get_statement atď.

Komunikácia Backendu s Frontendom

Základnou dátovou štruktúrou celého projektu je Class **Statement** ktorý má všetky potrebné atribúty ako napríklad id, description, proof…, táto trieda obsahuje aj veľmi dôležitý konštruktor ktorému ak dáme potrebné json dáta, tak ich vytiahne a priradí ich jednotlivým atribútom Statementu.

Na komunikáciu s Backendom sme vo Frontende vytvorili class **BackendAdapter** ktorý implementuje potrebné metódy na prácu s Backendom ako napríklad parseFile, getStatement a proccessBatch ktorá vracia viacero statementov naraz. Tieto metódy vracajú class Statement.

Fronend

Frontend je implementovaný pomocou čistého JavaScriptu bez použitia frameworkov. Funkcionalita aplikácie je rozdelená do troch hlavných častí, každá s vlastnou stránkou.

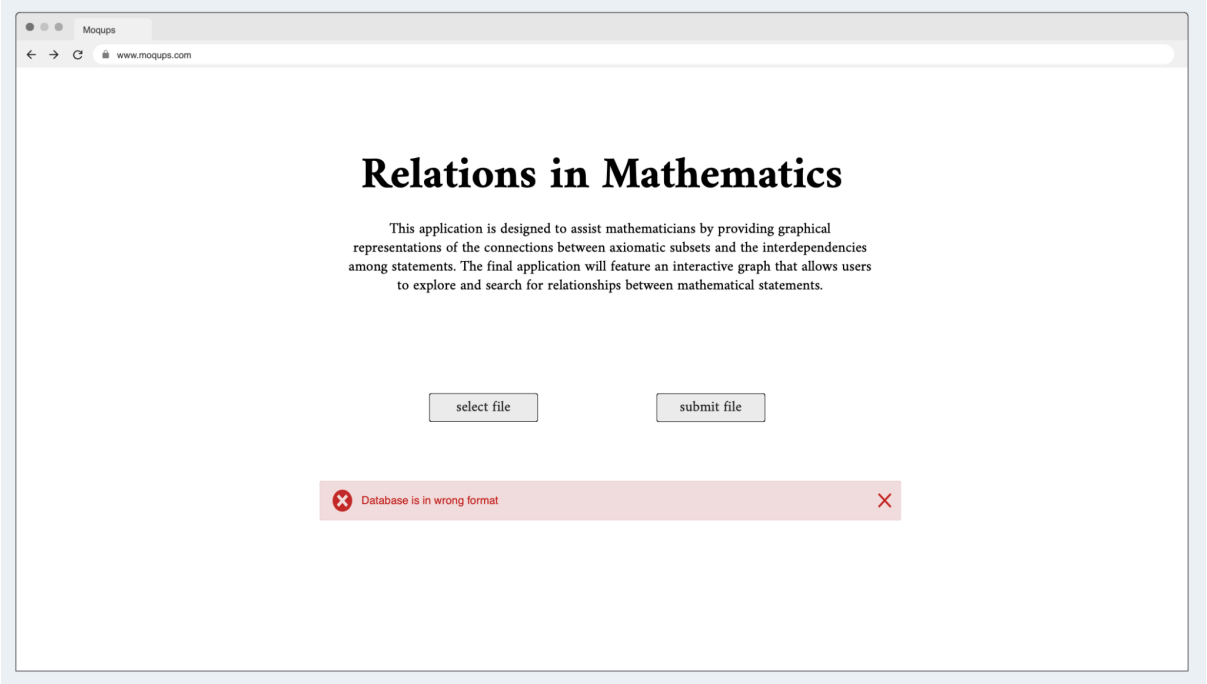
Welcome Page je implementovaná v class WelcomeApp ktorá dovoľuje uploadnuť .mm súbor na čo používa class FileUploadHandler. Na animácie a funkcionalitu grafických prvkov je použitá class UIStateManager.

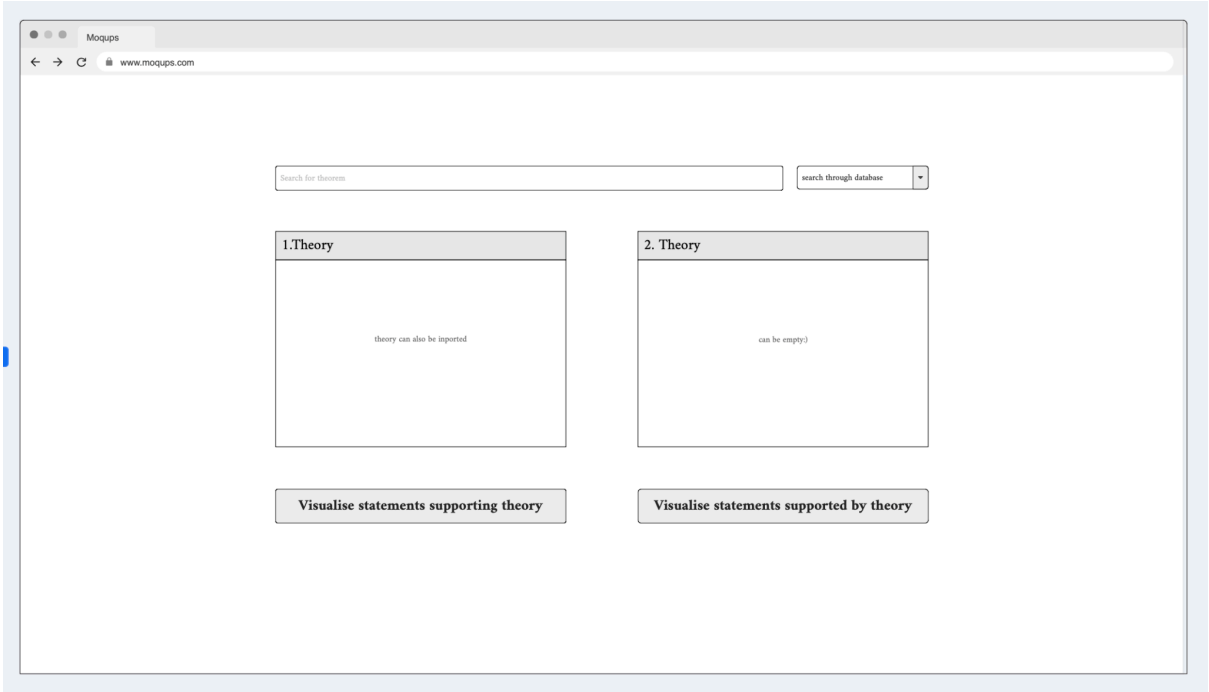
Theory Page je implementovaná v class TheoryApp. Umožňuje vytvoriť dve teorie pridavanim statementov pomocou vyhladavania. Na searchovanie používa search_statement metódu. Na prácu s teóriami používa TheoryHandler ktorý ukladá id statementov v dvoch teóriach.

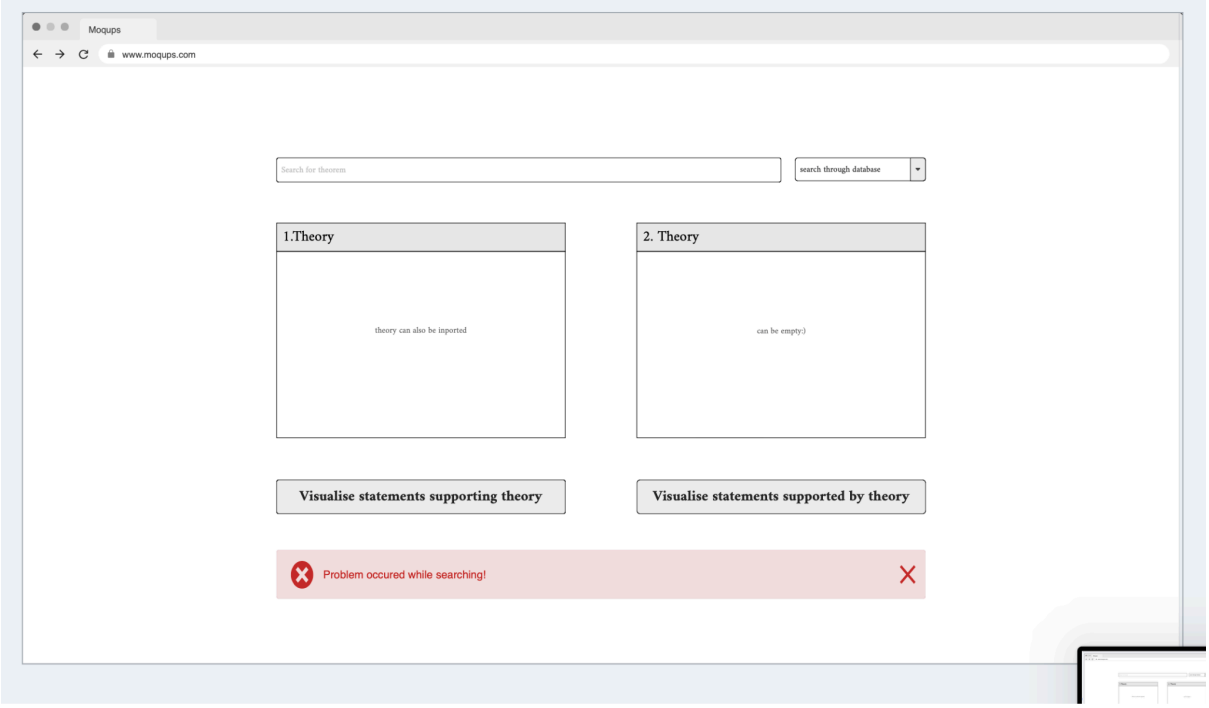
Searchovací algoritmus je implementovaný v Backende v DataHandler.py (metóda findStatements) pomocou custom implementace algoritmu Fuzzy Search.

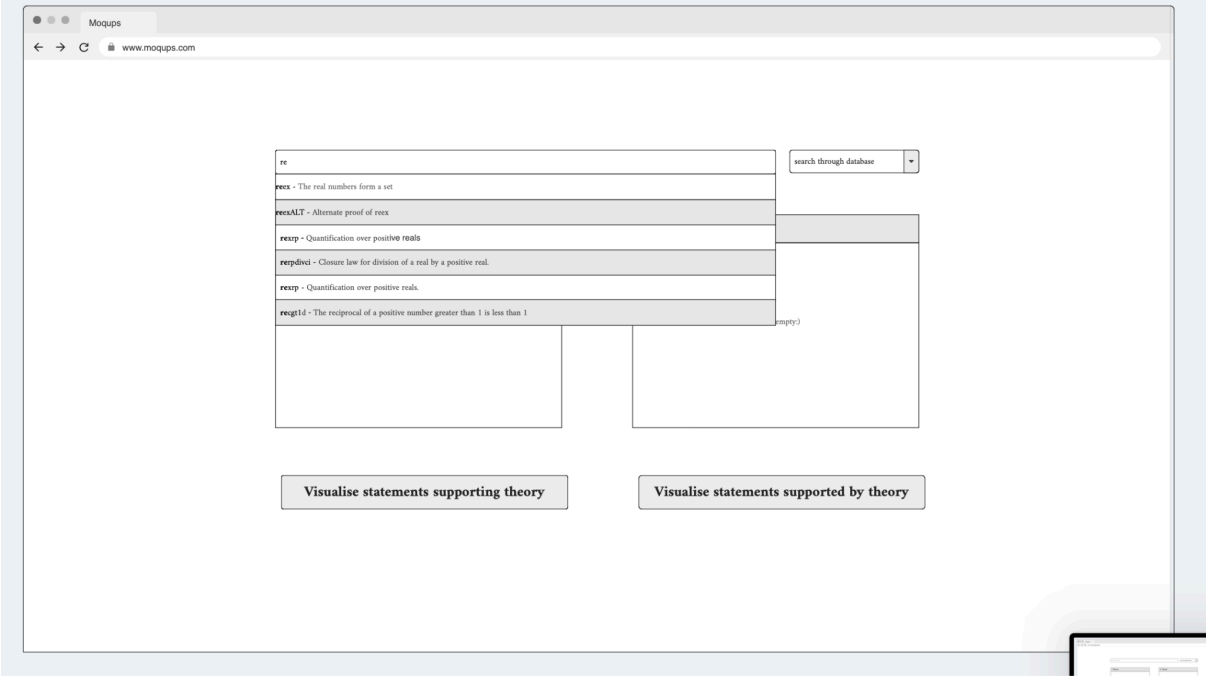
Graph Page je implementovaná v class GraphMaster. Umožňuje vizualizácie závislostí medzi statementami pomocou knižnice **Cytoscape.js** z tej používame metódy nodes(), fit(), zoom(), batch(), layout(), on() a run(). Volbu zobratenia sme implementovali v class Settings. Táto stránka používa statementy z class TheoryHandler ktoré boli zvolené v Theory Page, na prenos dát medzi dvoma stránkami bolo do TheoryHandler potrebné implementovať metódy saveTheories a loadTheories.

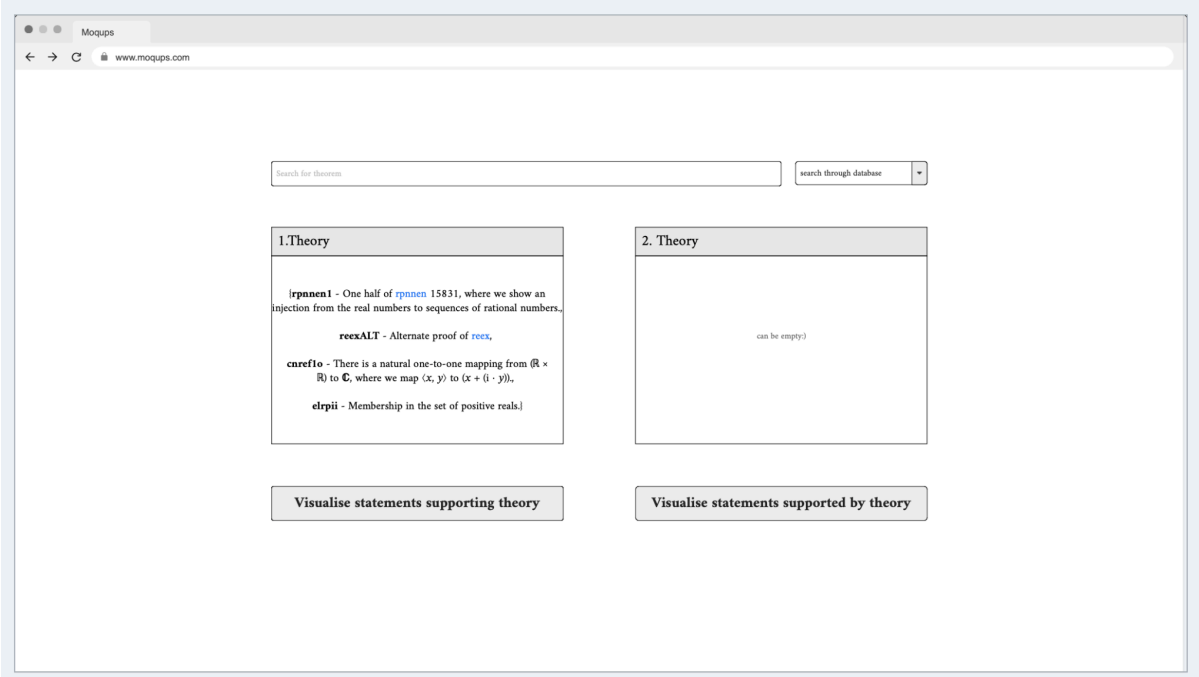
3. Používateľské rozhranie

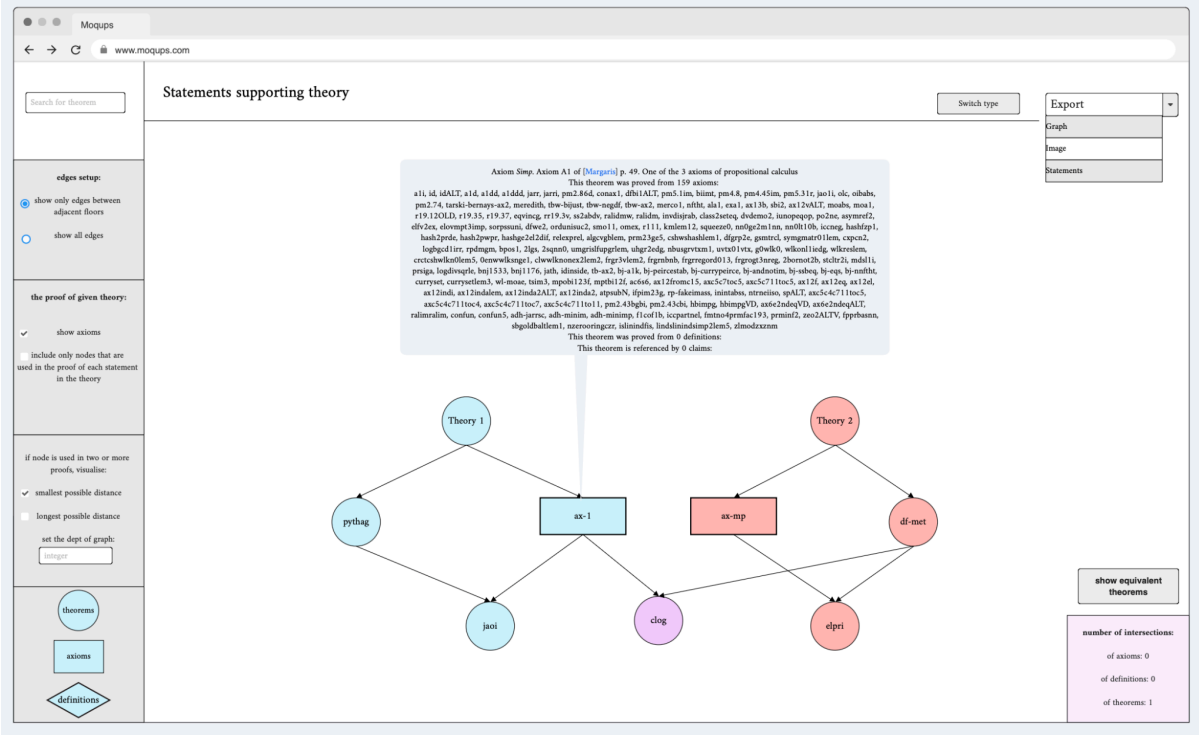


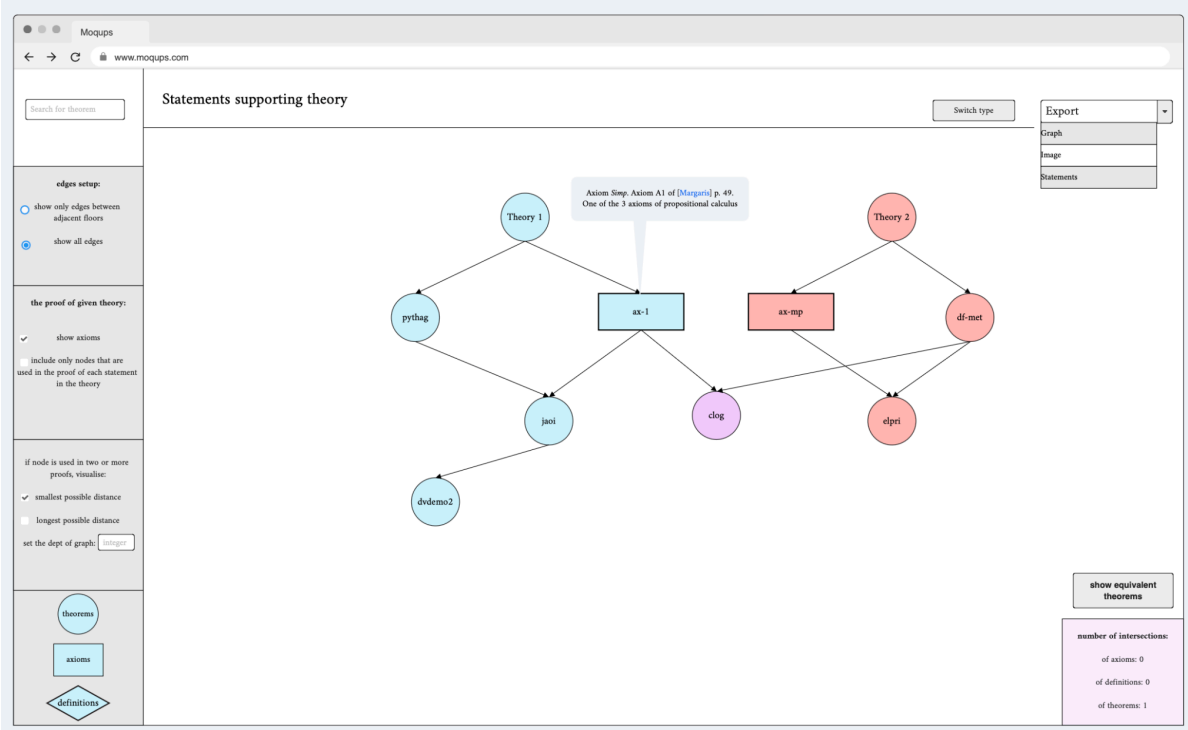


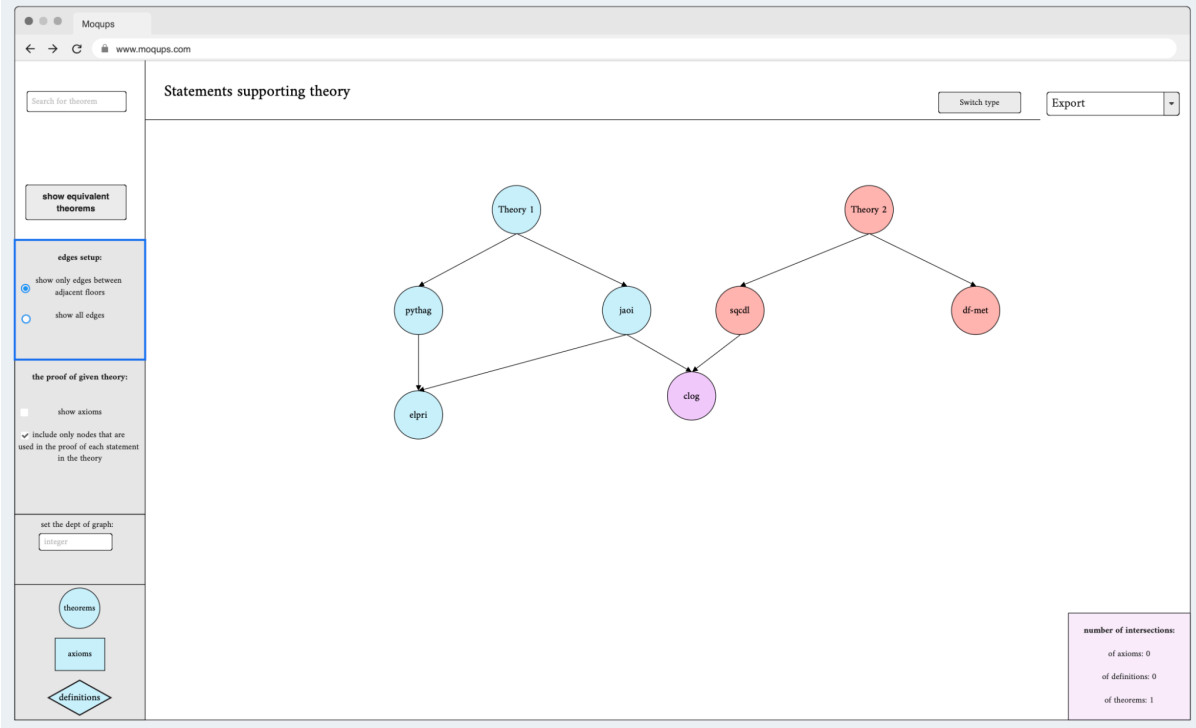

Moqups
www.moqups.com
Search for theorem
search through database
1.Theory
theory can also be inported
2. Theory
can be empty:)
Visualise statements supporting theory
Visualise statements supported by theory
Problem occured while searching!

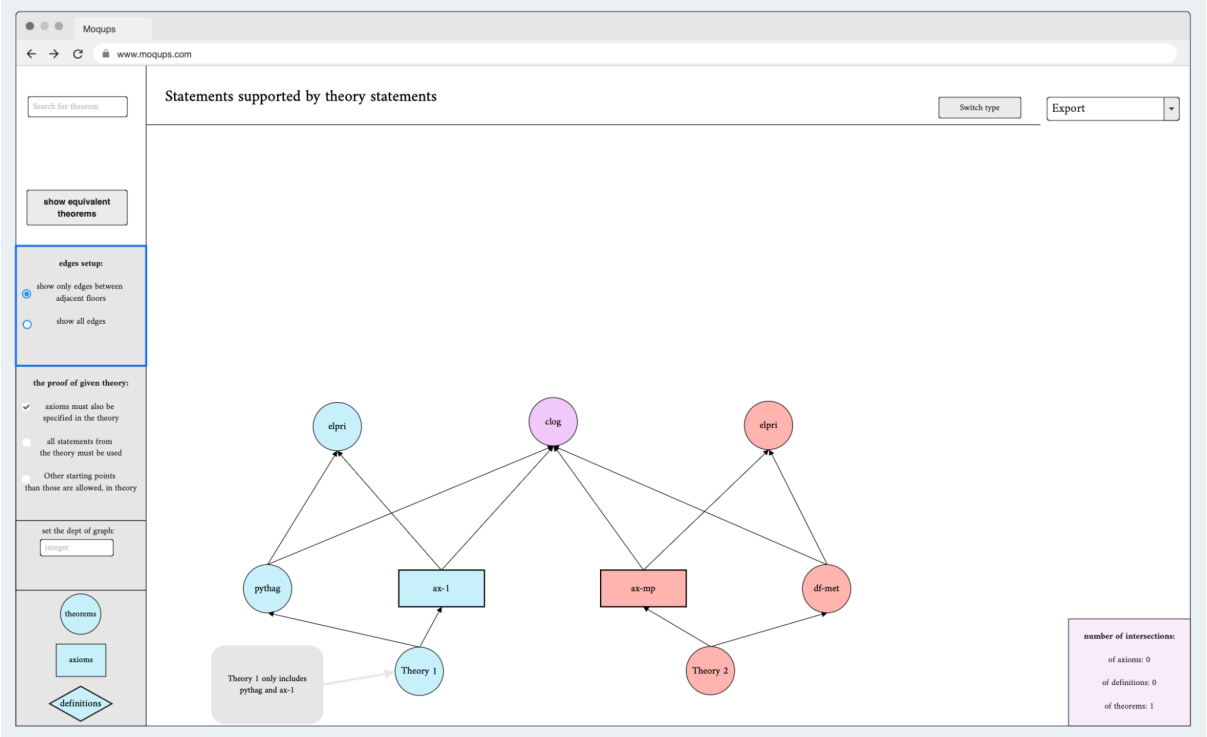

Moqups
www.moqups.com
re
search through database
reex - The real numbers form a set
reexALT - Alternate proof of reex
rexrp - Quantification over positive reals
rerpdivcl - Closure law for division of a real by a positive real.
rexrp - Quantification over positive reals.
recgt1d - The reciprocal of a positive number greater than 1 is less than 1
empty:)
Visualise statements supporting theory
Visualise statements supported by theory

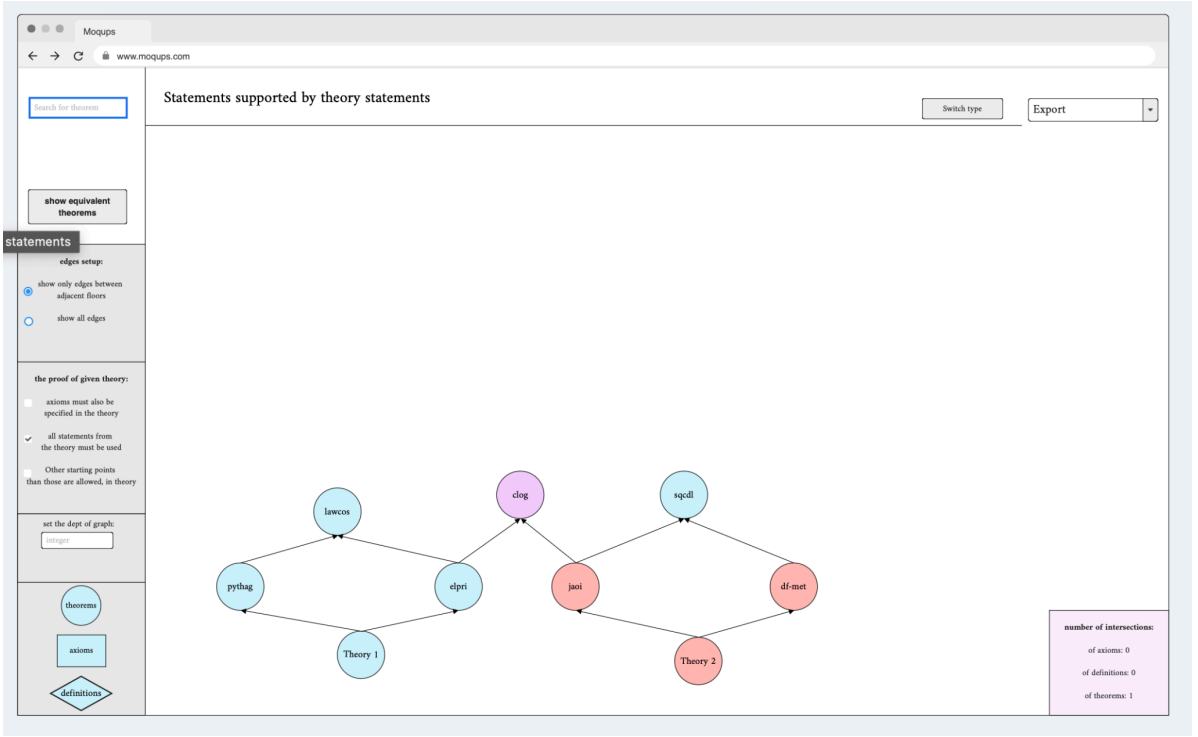

Moqups
www.moqups.com
Search for theorem
search through database
1.Theory
{rpnnen1 - One half of rpnnen 15831, where we show an injection from the real numbers to sequences of rational numbers.,
reexALT - Alternate proof of reex,
cnref1o - There is a natural one-to-one mapping from (ℝ × ℝ) to ℂ, where we map ⟨x, y⟩ to (x + (i · y)),
elrpii - Membership in the set of positive reals.}
2. Theory
can be empty:)
Visualise statements supporting theory
Visualise statements supported by theory

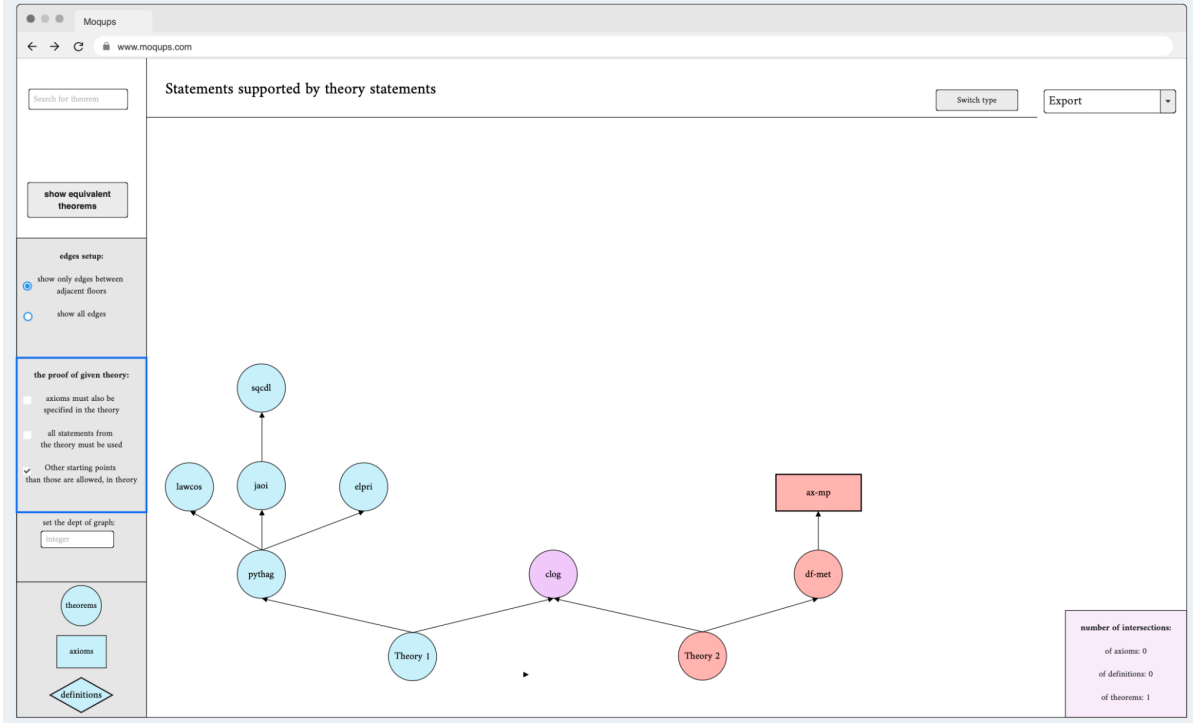

Moqups
www.moqups.com
Search for theorem
Statements supporting theory
Switch type
Export
Graph
Image
Statements
edges setup:
show only edges between adjacent floors
show all edges
the proof of given theory:
show axioms
include only nodes that are used in the proof of each statement in the theory
if node is used in two or more proofs, visualise:
smallest possible distance
longest possible distance
set the dept of graph:
integer
theorems
axioms
definitions
Axiom Simp. Axiom A1 of [Margaris] p. 49. One of the 3 axioms of propositional calculus
This theorem was proved from 159 axioms:
This theorem was proved from 0 definitions:
This theorem is referenced by 0 claims:
Theory 1
Theory 2
pythag
ax-1
ax-mp
df-met
jaoi
clog
elpri
show equivalent theorems
number of intersections:
of axioms: 0
of definitions: 0
of theorems: 1


Moqups
www.moqups.com
Search for theorem
Statements supporting theory
Switch type
Export
Graph
Image
Statements
edges setup:
show only edges between adjacent floors
show all edges
the proof of given theory:
show axioms
include only nodes that are used in the proof of each statement in the theory
if node is used in two or more proofs, visualise:
smallest possible distance
longest possible distance
set the dept of graph:
integer
theorems
axioms
definitions
Axiom Simp. Axiom A1 of [Margaris] p. 49. One of the 3 axioms of propositional calculus
Theory 1
Theory 2
pythag
ax-1
ax-mp
df-met
jaoi
clog
elpri
dvdemo2
show equivalent theorems
number of intersections:
of axioms: 0
of definitions: 0
of theorems: 1


Moqups
www.moqups.com
Search for theorem
Statements supporting theory
Switch type
Export
show equivalent theorems
edges setup:
show only edges between adjacent floors
show all edges
the proof of given theory:
show axioms
include only nodes that are used in the proof of each statement in the theory
set the dept of graph:
integer
theorems
axioms
definitions
Theory 1
Theory 2
pythag
jaoi
sqcdl
df-met
clog
elpri
number of intersections:
of axioms: 0
of definitions: 0
of theorems: 1

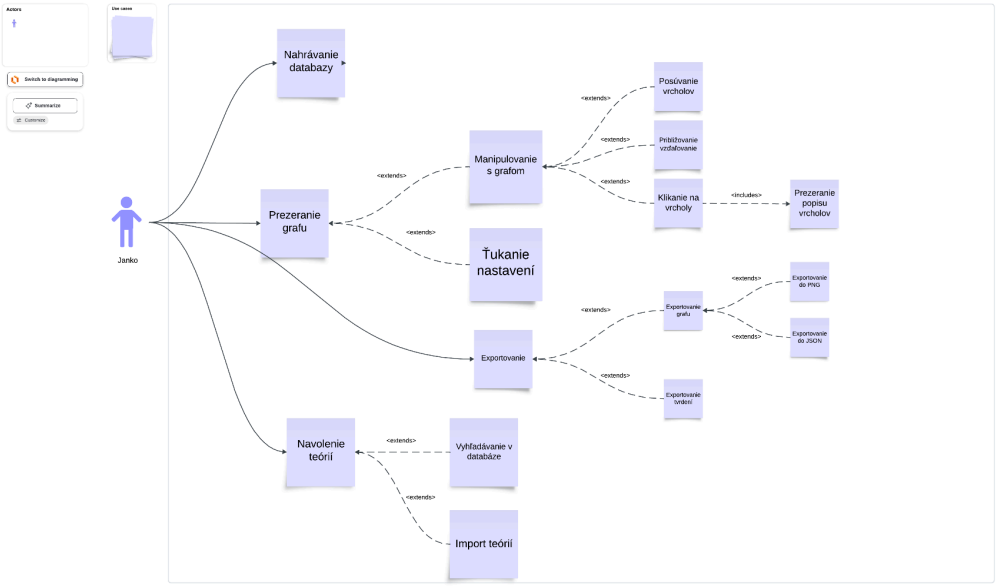

Moqups
www.moqups.com
Search for theorem
Statements supported by theory statements
Switch type
Export
show equivalent theorems
edges setup:
show only edges between adjacent floors
show all edges
the proof of given theory:
axioms must also be specified in the theory
all statements from the theory must be used
Other starting points than those are allowed, in theory
set the dept of graph:
integer
theorems
axioms
definitions
elpri
clog
elpri
pythag
ax-1
ax-mp
df-met
Theory 1 only includes pythag and ax-1
Theory 1
Theory 2
number of intersections:
of axioms: 0
of definitions: 0
of theorems: 1


Moqups
www.moqups.com
Search for theorem
Statements supported by theory statements
Switch type
Export
show equivalent theorems
statements
edges setup:
show only edges between adjacent floors
show all edges
the proof of given theory:
axioms must also be specified in the theory
all statements from the theory must be used
Other starting points than those are allowed, in theory
set the dept of graph:
integer
theorems
axioms
definitions
lawcos
clog
sqcdl
pythag
elpri
jaoi
df-met
Theory 1
Theory 2
number of intersections:
of axioms: 0
of definitions: 0
of theorems: 1

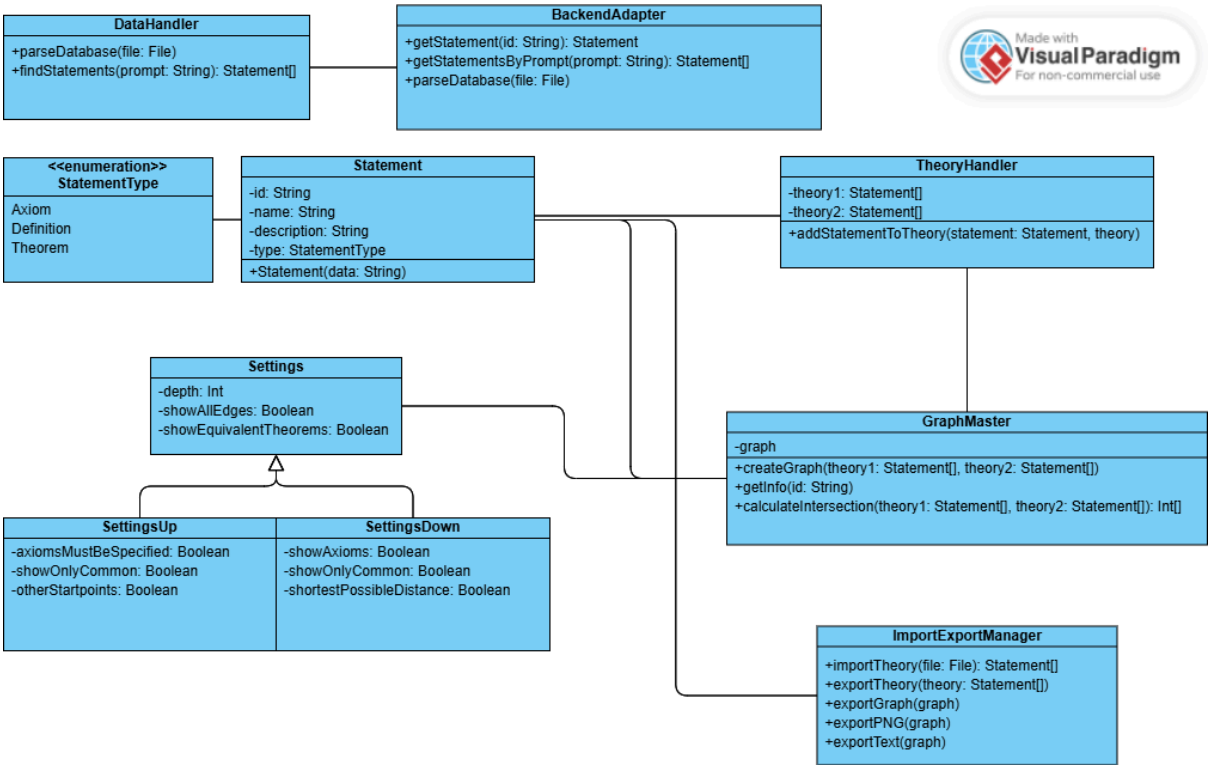

Moqups
www.moqups.com
Search for theorem
Statements supported by theory statements
Switch type
Export
show equivalent theorems
edges setup:
show only edges between adjacent floors
show all edges
the proof of given theory:
axioms must also be specified in the theory
all statements from the theory must be used
Other starting points than those are allowed, in theory
set the dept of graph:
integer
theorems
axioms
definitions
sqcdl
lawcos
jaoi
elpri
ax-mp
pythag
clog
df-met
Theory 1
Theory 2
number of intersections:
of axioms: 0
of definitions: 0
of theorems: 1

4. UML diagramy

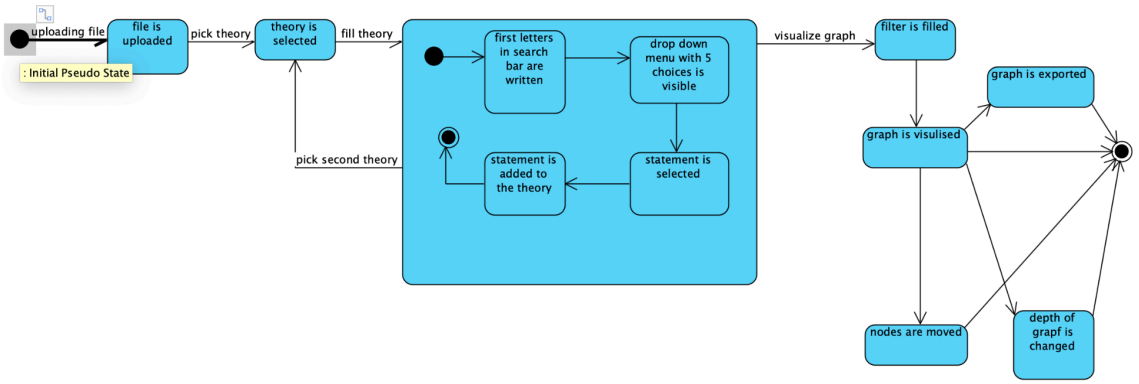
4.1 Use-case diagram



4.2 Class diagram



4.3 Stavový diagram



5. Syntax .mm súboru

Súbor .mm v je textový súbor obsahujúci databázu formálneho matematického systému. Je štrukturovaný v jazyku Metamath a obsahuje informácie o axiómach, definíciách a teoremoch s ich dôkazmi.

Štruktúra tohto súboru je pekne opísaná na tejto stránke Introduction to Metamath alebo podrobne rozpísaná v metamath dokumentácii metamath.pdf v kapitole 4 na strane 111.

6. Špecifikácia import/export súborov

Používateľ má možnosť exportovať 3 typy súborov:

1. Exportovanie obrázku grafu - v štandardnom PNG formáte.
2. Exportovanie grafu - súbor bude v GraphML formáte (.graphml), viac info na GraphML Format
3. Exportovanie statementov - textový súbor obsahujúci informácie o tom aké statementy sú v teóriach. Súbor obsahuje dva riadky, v prvom riadku sú id-čka statementov v teórii 1 oddelené medzerou, v druhom riadku obdobne ale statementy pochádzaju z teórie 2

Súbor ktorý používateľ môže importovať je textový súbor s rovnakou štruktúrou ako súbor opísaný v bode 3.

7. Plán implementácie

- Backend
  - DataHandler
    - ParseDatabase(file_path)
    - FindStatements(prompt)
- Frontend
  - BackendAdapter
    - Statement getStatement(id)
    - Statement[] getStatementsByPrompt(prompt)
    - parseDatabase(file)
  - Statement
    - Statement(json data)
    - id
    - name
    - description
    - …
    - type (axiom, definition, theorem)
  - TheoryHandler
    - Statement[] theory1, theory2
    - AddStatementToTheory(statement, theory)
  - Settings
    - int depth
    - bool showAllEdges
    - bool showEquivalentTheorems
  - SettingsUp : Settings
    - bool axiomsMustBeSpecified
    - bool showOnlyCommon
    - bool otherStartpoints
  - SettingsDown : Settings
    - bool showAxioms
    - bool showOnlyCommon
    - bool shortestPossibleDistance
  - GraphMaster
    - graph
    - CreateGraph(theory1, theory2)
    - GetInfo(id)
    - int[] CalculateIntersection(theory1, theory2)
  - ImportExportManager
    - Statement[] ImportTheory(file)
    - ExportTheory(theory)
    - ExportGraph(graph)
    - ExportPNG(graph)
    - ExportText(graph)

- Poradie implementácie

- ☑ ~~Create empty classes~~
- ☑ ~~DataHandler~~
- ☑ ~~Statement~~
- ☑ ~~BackendAdapter~~
- ☑ ~~HTML/CSS/JS files~~
- ☑ ~~TheoryHandler~~
- ☑ ~~Settings~~
- ☑ ~~GraphMaster~~
- ☑ ~~ImportExportManager~~

8. Testovacie scenáre

Scenár 1, Startup:

- Užívateľ spustil webovú aplikáciu cez index.html, a nahrá mm súbor.
    - ak je súbor v zlom formáte systém vypíše error message
- Používateľ stlačí submit button.
    - systém zobrazí ďalšiu “stránku” aplikácie s názvom search

Scenár 1 pokrýva kapitolu 3.1 Application startup na 100%

Scenár 2, Vytvorenie teórie:

- vykonáme Startup a v search bare vyhľadáme, tvrdenie.
    - pokiaľ nastane v priebehu hľadania problém, dostaneme error message.
    - inak nám zobrazí ponuku prvých 6 tvrdení s najlepšov zhodou.
- Takto si vyklikáme obe teórie a stlačíme tlačidlo Visualise.
    - systém spracuje dané teórie a vytvorí graf, zobrazí ďalšiu stránku aplikácie s názvom Statements supporting theory, kde následne zobrazí vytvorený graf. (prienik dvoch teórií je zvýraznený farebne)

Scenár 2 pokrýva kapitolu 3.2.1 Add statements na 100% (a ešte iné kapitoly z časti)

Scenár 3, Práca s grafom 1:

- Vykonáme Vytvorenie teórie, následne si zaklikneme checkboxy/radiobuttony
    - graf sa podľa zakliknutých checkboxov interaktívne zmení.
- Môžeme približovať/vzdaľovať, hýbať sa po plátne a hýbať vrcholmi grafu
    - interaktívne zobrazenie
- Prejdeme myšou nad vrchol
    - zobrazí nám meno vrcholu
- Klikneme na daný vrchol
    - zobrazí nám podrobné informácie o vrchole
- Prejdeme myšou nad iný vrchol
    - zobrazí nám vzdialenosť od daného vrcholu

Scenár 3 pokrýva kapitolu 3.4 Manipulation with graph na 100% a zvyšok kapitoly 3.2 Search

Scenár 4, Práca s grafom 2:

- Vykonáme Vytvorenie teórie, následne si zaklikneme tlačidlo switch type
    - graf zmení zobrazenie z “Statements supporting theory” na “Statements supported by theory statements”.
- Zaklikneme checkboxy/radiobuttony
    - graf sa podľa zakliknutých checkboxov interaktívne zmení.
- Nastavíme hĺbku grafu
    - graf zmení svoju hĺbku (zobrazí iba vrcholy siahajúce do danej hĺbky od teórie)

Scenár 3 pokrýva kapitolu 3.3 Graph visualisation na 100%

Scenár 5, Import a export:

- Vykonáme Práca s grafom 1, a stlačíme tlačidlo export
    - zrolujú sa nám možnosti exportu
- vyberieme postupne všetky 3 (Image, Graph, Statements)
    - downloadnú sa nám do počítača postupne png obrázok, graf v štandardnom graf formáte a dokument popisujúci všetky tvrdenia (nie na čítanie)
- Spravíme ďalšie zmeny (zmeníme hĺbku, zmeníme checkboxy)
    - graf zmení svoju hĺbku (zobrazí iba vrcholy siahajúce do danej hĺbky od teórie a prispôsobý svoj vzhľad)
- Importneme súbor Statements
    - zobrazí nám pôvodny graf pred vykonaním zmien

Scenár 3 pokrýva kapitolu 3.5 Graph visualisation na 100%