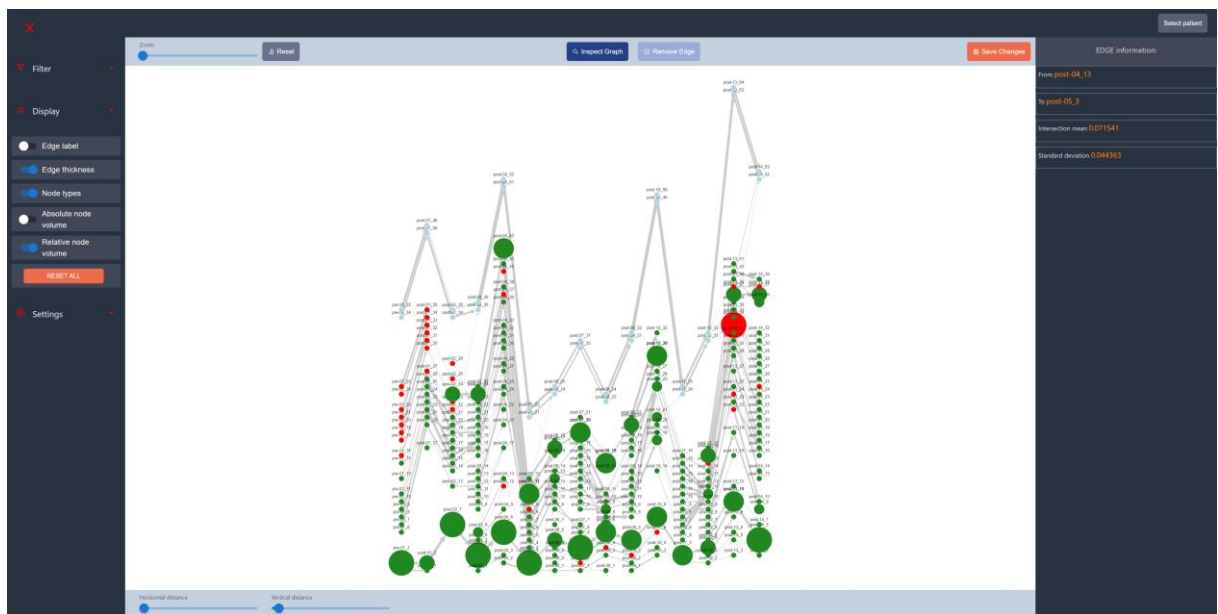


Bachelor Thesis 2022

Web-based visualization and review of longitudinal tumor mappings from medical image analysis



Student : Ekaterina Aymon

Professor : Adrien Depeursinge

Submitted on : 27th July 2022

SOURCE OF VISUAL ON THE TITLE PAGE

Image of the author.

ABSTRACT

Medical imaging plays an important role in tumor lesions detection and monitoring of treatment response. To observe the evolution of tumor lesions and thereby correct the treatment, the radiologist needs to compare the corresponding lesions across imaging examinations. However, such process of manual comparison of individual lesions is time-consuming and labor-intensive. To overcome this limitation, the automated process of identification of the corresponding lesions over time was introduced.

Referred as “longitudinal tumor mappings”, the automatically generated connections between tumor lesions must be manually reviewed and corrected in order to ensure accurate results. To facilitate these processes, this thesis introduces a web application, supporting the longitudinal mapping and review process. The developed web application is based on the approach of the representation of tumor mappings as a graph structure, where the nodes of the graph represent the tumor lesions, and the edges of the graph represent the longitudinal links between tumor lesions.

Based on the Cytoscape.js library and MERN stack technologies, the created application represents a full stack solution that allows visualization, manipulation, and persistence of the automatically constructed graphs of individual patients. The developed interface allows user to load an existing mapping graph, retrieve the information about nodes and edges, apply visual and filtering settings on the graph, correct the automatically generated connections between nodes and save an updated graph. The result of this study may be used for the further development of the longitudinal tumor mapping and review process.

Keywords: longitudinal mappings, medical imaging, radiomics, node-link diagram, graph visualization

FOREWORD

This bachelor thesis has been realized in the context of the bachelor's degree program in business information technology at the University of Applied Sciences Western Switzerland (HES-SO Valais-Wallis), in Sierre.

This work is carried out in close collaboration with the Computational Oncology group of the Precision Oncology Center at Lausanne University Hospital. The thesis aims to build a web application that supports the longitudinal mapping and review process.

This thesis is written based on the standards of the 6th edition of the American Psychological Association (APA).

ACKNOWLEDGEMENTS

I would like to express my gratitude to the people who guided and helped me throughout the realization of this work, concretely:

- Mr. Adrien Depeursinge, professor of the HES-SO, for the guidance, recommendations, and the complete support,
- Mr. Roger Schaer, scientific collaborator of the HES-SO, for the technical support during the project,
- Mr. Daniel Abler, scientific collaborator of the HES-SO, for helping to understand the medical data,
- Mr. Michel Cuendet, head of research of the Precision Oncology Center in Lausanne University Hospital, for useful suggestions about the representation of the application,
- All team, for the good relation, availability and openness.

TABLE OF CONTENTS

LIST OF TABLES	VIII
LIST OF FIGURES	IX
LIST OF ABBREVIATIONS.....	XI
INTRODUCTION	1
CONTEXT AND RESEARCH PROBLEM	1
GOAL AND OBJECTIVES	4
THESIS PHASES	5
1. METHODOLOGY.....	6
1.1. GRAPH STRUCTURE.....	6
1.2. GRAPH FILE FORMATS	9
1.2.1. CSV.....	9
1.2.2. GraphML.....	10
1.2.3. GML	10
1.2.4. JSON.....	11
1.3. GRAPH VISUALIZATION LIBRARIES.....	12
1.3.1. General overview.....	12
1.3.2. Requirements	13
1.3.3. Python graph visualization libraries	14
1.3.4. JavaScript graph visualization libraries	15
1.3.5. Graph libraries comparison.....	16
1.4. USE CASES DEFINITION	17
1.5. APPLICATION ARCHITECTURE.....	19
2. RESULTS.....	23
2.1. GENERAL OVERVIEW	23
2.2. GRAPH DATA REPRESENTATION	25
2.3. GRAPH VISUALIZATION	28
2.3.1. Graph layout	28
2.3.2. Visual features.....	31

2.3.3. <i>Filtering features</i>	36
2.3.4. <i>Graph interaction</i>	38
2.4. GRAPH MANIPULATION	41
2.5. GRAPH STORAGE	42
3. DISCUSSION	45
3.1. ANALYSIS OF THE RESULTS	45
3.2. LIMITATIONS	45
3.3. FUTURE WORK	46
4. CONCLUSION	49
REFERENCES	50
APPENDIX I: COMPARATIVE ANALYSIS OF GRAPH VISUALIZATION LIBRARIES	53
APPENDIX II: ALL COMPLETED USER STORIES IN THE PRODUCT BACKLOG	54
APPENDIX III: EXAMPLE OF SPRINT BACKLOG	55
AUTHOR'S DECLARATION	56

LIST OF TABLES

Table 1: The API methods of Express.js	43
--	----

LIST OF FIGURES

Figure 1: Example of automatically established longitudinal mappings that link corresponding tumor lesions across three imaging examinations	6
Figure 2: Mockup showing a longitudinal tumor mapping graph	8
Figure 3: A comparison of layout techniques, utilizing the same information. Radial layout (in the left) and force-directed layout (in the right).....	12
Figure 4: Use Cases of tumor mapping application	17
Figure 5: Development phases of tumor mapping application based on use cases.....	18
Figure 6: MERN stack components.....	20
Figure 7: MERN stack three-tier architecture	21
Figure 8: Three-tier architecture of tumor mapping application	22
Figure 9: Input form of the home page	23
Figure 10: Main page of the application	24
Figure 11: Open sidebar (in the left) and open submenus of sidebar (in the right)	25
Figure 12: Node structure of tumor mapping graph	26
Figure 13: Edge structure of tumor mapping graph	27
Figure 14: Node position changing algorithm	29
Figure 15: Time-oriented graph layout	29
Figure 16: Basic node distance (in the left) and increased distance (in the right)	30
Figure 17: Example of enabled “edge label” feature	31
Figure 18: Example of enabled “edge thickness” feature	31
Figure 19: Example of enabled “node types” feature.....	32
Figure 20: Example of enabled “absolute node volume” feature.....	33
Figure 21: Relative node volume algorithm	33
Figure 22: Example of enabled “relative node volume” feature.....	34
Figure 23: Comparison between absolute (in the left) and relative (in the right) node volume options	35
Figure 24: Configuration panel of visual features	35
Figure 25: Node type filtering: basic graph (in the left) and filtered by suspicious node type graph (in the right)	36
Figure 26: Threshold filtering: threshold filter set to 0 (in the left) and to 0.5 (in the right) ..	37
Figure 27: Configuration panel of filtering.....	37
Figure 28: Examples of node and edge information	38

Figure 29: Graphical representation of all related ROIs on element hovering	39
Figure 30: Basic subgraph (top image) and three new subgraphs after threshold value filtering (bottom image)	40
Figure 31: Settings menu	40
Figure 32: Example of hovering over edge	41
Figure 33: Example of one element of the Patients collection in MongoDB	42
Figure 34: Mongoose schema definition	43
Figure 35: Fetch graph function	44
Figure 36: Update graph function	44

LIST OF ABBREVIATIONS

API	Application Programming Interface
CSR	Client-Side Rendering
CSS	Cascading Style Sheets
CSV	Comma-Separated Values
CT	Computed Tomography
GML	Graph Modelling Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identifier
IT	Information Technology
JSON	JavaScript Object Notation
MRI	Magnetic Resonance Imaging
PET	Positron Emission Tomography
ROI	Region Of Interest
SSR	Server-Side Rendering
SVG	Scalable Vector Graphics
WebGL	Web Graphics Library
XML	Extensible Markup Language

INTRODUCTION

Context and research problem

Medical imaging plays an important role in the tumor detection and monitoring. Early detection of cancer based on imaging is probably the major reason of the reduction in mortality for certain cancers (Fass, 2008, p. 115). At the same time, medical imaging provides the information about tumor lesion treatment response, which is used for its correction or its further observation.

Traditional radiographic imaging evaluation of tumors is based on qualitative features, such as tumor density, pattern of enhancement, regularity of tumor margins and other characteristics (Bi, Hosny, Schabath, Giger, & Birkbak, 2019, p. 128). In comparison, another approach, called radiomics, extracts a large number of quantitative features from computed tomography (CT), magnetic resonance imaging (MRI), or positron emission tomography (PET) images and convert it into mineable high-dimensional data (Gillies, Kinahan, & Hricak, 2016, p. 564). Radiomics “look at individual differences that exist in cancer cells in order to determine a personalized, highly targeted treatment course, solutions that classical methods cannot provide” (Meng, et al., 2019, p. 10851).

According to Meng et al. (2019, p. 10852), the process of radiomics consists of six phases: (1) acquisition of image data, (2) calibration of tumor regions, (3) segmentation of tumor regions, (4) extraction and quantification of features, (5) image database establishment, and (6) classification and prediction. The automation of these phases is expected to have a positive impact in radiology and to improve clinical decision making.

One of the main challenges facing researchers for automating this process is the automation of tumor segmentation. The segmentation represents the correct detection of spatial location of a tumor, called region of interest (ROI), that can be used for the different purposes, for example, to compute imaging biomarkers (including radiomics features) for response assessment (Moreau, et al., 2022, p. 2). The manual segmentation performed by the expert remains the gold standard. However, the manual segmentation is time-consuming and labor-intensive (Moreau, et al., 2022). The automation of the tumor segmentation is challenging firstly because many tumors have indistinct borders (Gillies, Kinahan, & Hricak, 2016, p. 568). Secondly, the segmentation is affected from intrinsic and extrinsic factors,

including spatial resolution and noise of the image, as well as shape, texture, and location of pathologies (Ha, Choi, Paeng, & Cheon, 2019, p.17).

Recent study of Capobianco et al. (2021, p. 31) presents an automated segmentation algorithm of the identification of all 3-dimensional regions of interest (ROIs) with increased tracer uptake. This method allows to define whether each ROI is classified as non-suspicious (benign) or suspicious (malignant) uptake. However, the expert still needs to supervise and correct the potential incorrect classification of the regions of interest (Capobianco, et al., 2021, p. 35).

Second big challenge represents the automatic identification of the corresponding tumor lesions on multiple timepoints. This, what we call “longitudinal tumor mappings”, allows to trace the evolution of individual tumor lesions over time and provides valuable information about patient health situation, that can be used to monitor and correct treatment. For example, Basler et al. (2020, p. 4416) uses longitudinal individual lesion analysis to investigate response assessment to the treatment. However, the lesions in that study are manually segmented at all timepoints. Matching lesions manually is time-consuming and labor-intensive, as an expert need to review multiple images and go back and forth between all these images for comparison (Cai, et al., 2021, p. 1). Therefore, the automation of this process could have a good impact on the performance. In the recent study of Cai et al. (2021, p.3), the automated process of tracking the lesion is presented, which could be used to improve the existing mechanisms of automated tracking.

Despite the existence of automating processes of tumor segmentation and longitudinal tumor mapping, the result of these processes should still be supervised and manually reviewed by the radiologist. The existing commercial software facilitate such manual review by an expert. For example, “MIM Software¹” and “mint Lesion^{TM2}” suggest to the expert the probable connected lesions of the different timepoints. Then, the expert needs to confirm or adjust the proposed connection between these lesions. However, the expert still must review every single lesion and relations of lesions between timepoints, which bring us again to the problem of time-consuming.

¹ Available at: <https://www.mimsoftware.com/> Accessed in June 2022

² Available at: <https://mint-medical.com/mint-lesion/clinical-routine> Accessed in June 2022

Another approach for the manual review of automatically defined tumor segmentation and longitudinal lesion mapping is proposed as a part of the study of the Computational Oncology group of the Precision Oncology Center at CHUV³, Lausanne University Hospital. This approach is based on the idea of the manual review of the entirety of automatically segmented and pre-mapped lesions instead of separate lesions and connections. For that purpose, the graph structure is introduced.

The graph structure consists of the nodes, representing ROIs that was identified from the image, and the edges, which represent potential connections between ROIs. Since the graph is a structure of the connected ROIs at different timepoints, the monitoring, modification and control of all connected ROIs and their links are greatly facilitated.

To the best of our knowledge, there is currently no web application that allows visualization and manipulation of automatically constructed graphs of the individual patients in the context of longitudinal mapping. The creation of such tool could facilitate the manual review of longitudinal tumor mappings as well as improve the automated mapping algorithm.

³ Centre hospitalier universitaire vaudois

Goal and Objectives

The goal of this thesis is to build a web application that allows visualization, manipulation, and persistence of the automatically constructed graphs of individual patients in order to facilitate the review and correction of longitudinal tumor mappings.

To achieve our goal, we have identified the following objectives:

- Define use cases that reflect the expert needs,
- Analyze current literature on graphs and highlight the main characteristics of graphs,
- Create mock-ups of possible user interface,
- Analyze and select the most appropriate library for graph visualization as well as select the appropriate file format for graph-structured data,
- Select an appropriate technology for web-based application as well as define the architecture of that application,
- Create a web-based application that allows visualization and manipulation of longitudinal tumor mapping graph.

Thesis phases

To begin with, we will consider what a graph is in general and how that structure can be used for the visualization of tumor mappings. We will look at existing file formats of graph-structured data and define the most relevant for our application.

Then, we will compare existing graph libraries and decide which ones are interested to us based on our requirements. We will transform all user suggestions into use cases, as well as select the appropriate architecture for the web application.

Afterwards, the results of the development of created web application will be described.

Finally, we will present our suggestions to possible future improvements of the current application.

1. METHODOLOGY

1.1. Graph structure

To achieve the goal set out in this study, we first need to explore what a graph structure is in general and what type of graph will be used to visualize the tumor mappings of the patient.

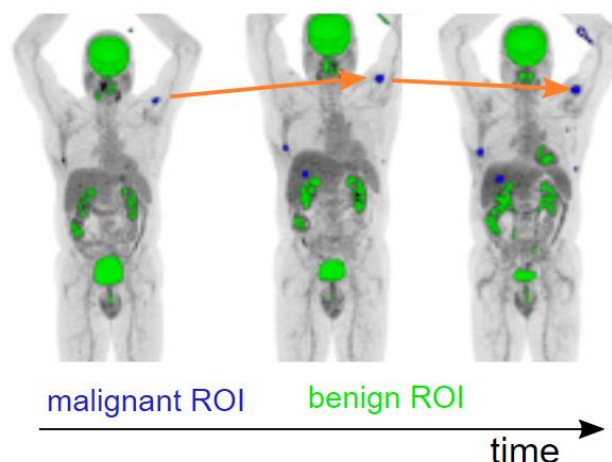
In general, any mathematical object involving points and connections between them may be called a graph (Gross, Yellen, & Zhang, 2013, p. 2). In the book “Graph Theory and Its Applications”, the definition of the graph is following:

A **graph** $G = (V, E)$ is a mathematical structure consisting of two finite sets V and E . The elements of V are called **vertices** (or **nodes**), and the elements of E are called **edges**.

Each edge has a set of one or two vertices associated to it, which are called its **endpoints** (Gross, Yellen, & Anderson, 2019, p. 2).

In our study, the nodes will represent the ROIs that have been identified from the medical images. The edges will represent automatically defined connections between these ROIs. Figure 1 shows an example of time-series PET/CT images, where the tumor lesions (blue) represent the malignant ROIs and the links (orange) represents the automatically established longitudinal mappings.

Figure 1: Example of automatically established longitudinal mappings that link corresponding tumor lesions across three imaging examinations



Source: image provided by Daniel Abler

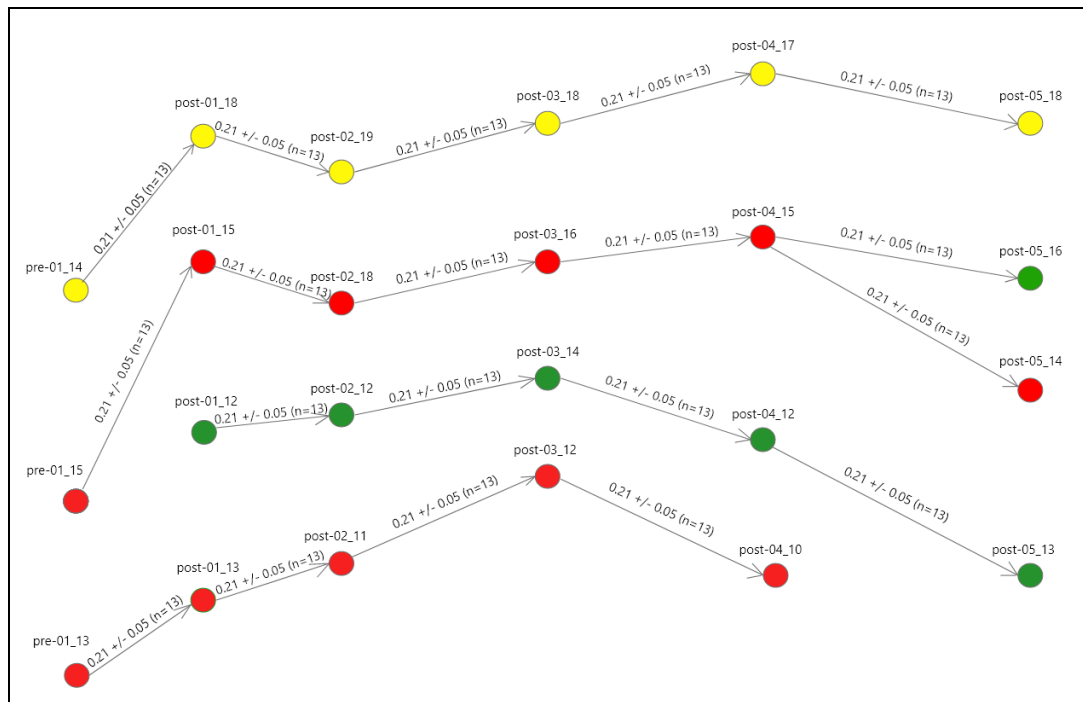
Since the sequence of nodes is considered as the observation of certain ROIs across time, the tumor mapping graph is defined as a path graph. Path graph is a graph where all of its nodes and edges lie on a single straight line (in opposite to cycle graph) (Gross, Yellen, & Anderson, 2019, p. 18). Another attribute of the graph that could demonstrate a time perspective is the direction of the edges, represented by the arrows. The source of the directed edge is designated as the tail and the target is designated as the head (Gross, Yellen, & Anderson, 2019, p. 3).

Label is also an important attribute of the graph. Even if for many problems only structural properties of a graph are required (Rahman, 2017, p. 23), labeling of nodes and edges could be apply to distinguish all elements. Since the tumor mapping graph will be manually reviewed by an expert, it is crucial to have labeled nodes and edges for the better graph visualization.

Last two important characteristics of the graph that could be considered are self-loop and multi-edge. A self-loop represents an edge that joins a node to itself. A multi-edge is a collection of two or more edges having identical endpoints (Gross, Yellen, & Anderson, 2019, p. 3). In regard to tumor mapping graph, it can be defined as a simple graph that has neither self-loops nor multi-edges.

Although the graph represents the entirety of automatically pre-contoured and pre-mapped lesions of one patient (e.g. full body), the focus of manual review of individual mappings should be confined to a certain part of the graph rather than the entire graph (Erciyas, 2021, p. 227). For this purpose, the definition of the subgraph is introduced. According to Gross, Yellen, & Anderson, "a subgraph of a graph G is a graph H whose vertices and edges are all in G " (2019, p. 74). In case of tumor mappings, we will consider a subgraph as a collection of all connected ROIs.

Figure 2 represents the general overview of the graph structure of longitudinal tumor mappings. As we can see, each node represents certain ROI in specific timepoint and the directed edges demonstrate the relations between ROIs. The graph in this mockup consists of four subgraphs that represent automatically established longitudinal mappings between ROIs.

Figure 2: Mockup showing a longitudinal tumor mapping graph

Source: author's source

With such graph representation, it becomes easy to observe the evolution of every single tumor lesion. For instance, the nodes “pre_01_13”, “post-01_13”, “post-02_11”, “post-03_12” and “post-04_10” in the bottom of the graph are considered as one tumor lesion, that was identified in five different timepoints. The labels of edges provide the information about average intersection of the ROIs.

The advantages of the graph structure for review of longitudinal tumor mappings are obvious. By reason of such data representation, the expert can observe the collection of all existing ROIs and their interrelations at different timepoints. Labels, colors and other additional attributes assist to distinguish specific elements. Thereby, the general overview of patient's lesions allows to rapidly define the lesion trajectories that should be revised.

We have considered what a graph is, what types it has and how it can be applied concerning the longitudinal tumor mappings. In the next chapters we will compare the data format for the graph structure as well as the libraries for graph visualization.

1.2. Graph file formats

In order to create, modify and store the graph of longitudinal mappings, the graph should be recorded in specific file format. An appropriate graph file format may simplify the process of data preparation and data retrieving.

There are several standard file formats for graphs. Graph files have a specific structure that allows the transformation of raw data into set of nodes and set of edges. In this chapter, we will consider the most popular formats since it is important for further graph visualization.

1.2.1. CSV

By definition, a comma-separated value (CSV) file format is a format in which tabular data (numbers and text) is encoded as plain text and the fields of the table are separated by commas (or any predefined separator character) (Butterfield, Ngondi, & Kerr, 2016). Rows in the table correspond to lines in the file. A CSV has a .csv file extension.

The CSV example below represents a graph with 3 edges: “a” -> “b”, “b” -> “c” and “b” -> “d” (Gephi, 2022) :

```
a;b
b;c;d
```

More complex examples of CSV files contain header row and additional data:

Nodes.csv:	Edges.csv:
Label, Volume	Source, Target, Label
"Node1", 100	"Node1", "Node2", "Edge1"
"Node2", 200	"Node2", "Node4", "Edge2"
"Node4", 300	"Node4", "Node12", "Edge3"
"Node12", 50	

The advantage of CSV file is that it can be easily opened, edited and exported. The disadvantage consists of low data quality control (Brath & Jonker, 2015).

1.2.2. GraphML

The GraphML file format uses .graphml extension and has a XML structure. It supports attributes for nodes and edges and hierarchical graphs (Gephi, 2022).

Basic construction of graph consists of the header with XML Schema reference and the graph body. An example of the graph body is below (Brandes, Eiglsperger, & Lerner, s.d.):

```
<graph id="G" edgedefault="directed">
  <node id="n0"/>
  <node id="n1"/>
  <edge id="e1" source="n0" target="n1"/>
</graph>
```

It is possible to set the attributes to nodes or edges:

```
<key id="d1" for="node" attr.name="color" attr.type="string"/>
```

And then specify a value in the element:

```
<node id="n1">
  <data key="d1">green</data>
</node>
```

The main advantages of this graph format are standard syntax, based on XML and support of many features such as graph attributes or references to external data.

1.2.3. GML

The Graph Modelling Language (GML) is a file format which consists of hierarchical key-value lists. An example of GML format is below:

```
graph [
  comment "My graph"
  directed 1
  node [id 1 label "Node 1" volume 50]
  node [id 2 label "Node 2" volume 20]
  edge [source 1 target 2 label "Edge from node 1 to node 2"]
]
```

The advantages of such graph format are its flexibility, extensibility and simple syntax. The main disadvantage is that GML does not identify data type (Brath & Jonker, 2015).

1.2.4. JSON

The JavaScript Object Notation (JSON) is a lightweight data-interchange format that is completely language independent. JSON file format is often used with web-based interactive visualizations (Brath & Jonker, 2015).

The example of graph structure in JSON format:

```
{
  "nodes":[
    { "label":"Node1","volume":100 },
    { "label":"Node2","volume":200 },
    { "label":"Node4","volume":300 },
    { "label":"Node12","volume":50 }
  ],
  "edges":[
    { "source":"Node1","target":"Node2","value":1 },
    { "source":"Node2","target":"Node2","value":2 },
    { "source":"Node4","target":"Node12","value":4 }
  ]
}
```

The set of nodes and the set of edges use the list structure. Each individual element of node or edge is placed in curly braces and contains a set of attribute-value pair. The advantages of JSON file format are flexibility, compatibility and its universality.

We have covered the main file formats that are dedicated for graph data. Selection of the appropriate format affects how much data structure can be customized. At the same time, it is necessary to keep in mind the compatibility with the chosen graph visualization tool.

After careful analysis of existing formats, we came to the conclusion that JSON file format is an optimal format of graph data for our web application. It can be used with web-based interactive applications, it has a clear notation and it is easy to customize such graph data. In addition, it is simple to store JSON file in NoSQL⁴ Databases.

In the next chapter we will choose the appropriate graph visualization library for web-based application.

⁴ Not Only SQL database

1.3. Graph visualization libraries

This chapter is dedicated to selection of the most appropriate graph visualization library based on our requirements. We will define these requirements, conduct a comparative analysis of existing libraries and choose the optimal solution.

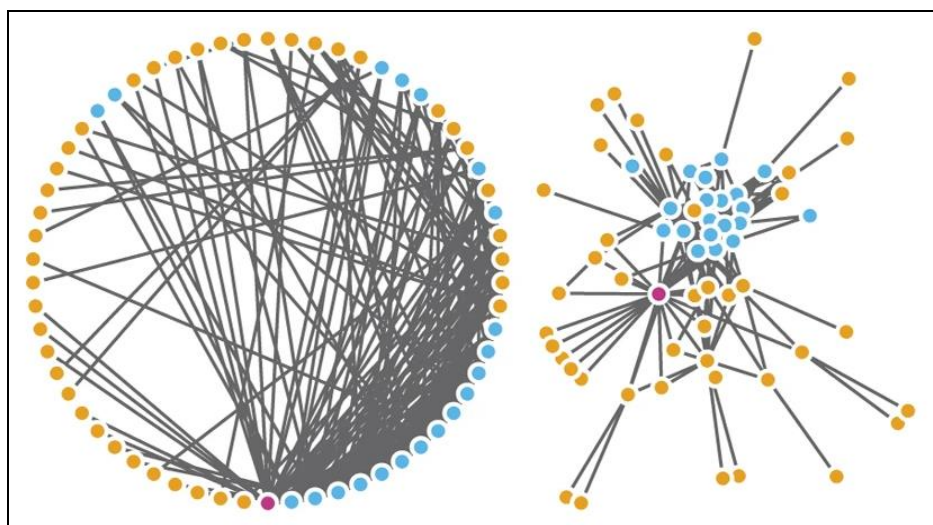
1.3.1. General overview

Among different types of visual representation of the graph, the node-link diagram is the most popular one (Han, Pan, Zhao, & Chen, 2021, p. 61). The process of the visualization of such diagram can be divided into several parts: data collection and cleaning, selection of the layout, adjusting of visual attributes and graph interaction (Brath & Jonker, 2015).

First step of visualization is data preparation. The data can exist in multiple sources or have some inconsistency, such as null or duplicate values. For these reasons, the data collection and cleaning should be performed. Then the cleaned data must be connected with the graph.

Second step of graph visualization is layout selection. Layouts are important because they give a sense of the graph structure. There are a large number of layouts, for example, radial and force-directed layouts (Figure 3). The time-oriented layout represents a good choice for the visualization of longitudinal tumor mappings as it facilitates perception of the sequence.

Figure 3: A comparison of layout techniques, utilizing the same information. Radial layout (in the left) and force-directed layout (in the right).



Source: Gehlenborg & Wong, 2012

Third step is attaching data to visual attributes to the graph elements. Visual attributes are significant for human perception because they highlight different aspects of the graph. Visual attributes include node attributes, edge attributes and labels. The examples of node attributes are color, size and shape. Edge attributes consist of color, weight and type of line. Label of a node or an edge represents an individual identity of an element.

Last step represents the graph interaction. It allows to explore the graph better, to see relevant details and to focus on certain parts of it. The main types of interactions that can be performed are: zoom, rotation, hover and click, filter, selection of subgraph, drag, move and remove of elements.

In general, it is possible to visualize a graph by three methods: creating own technology, using of end-user software or using of existing visualization libraries. We will only consider the last option, since the first will not fit the time frame of this study and the second one does not meet our requirements.

1.3.2. Requirements

To choose the most convenient library, we first need to define a list of requirements. Based on our study goal, we specified the following requirements:

- Free license, open source. For graph visualization, we search a free library that is supported by the developers and regularly updated.
- Web-based orientation. Even if we can find a free graph visualization end-user platforms, our focus is on the web-based libraries that allow a custom development.
- Graph customization. Node colors, edge thickness, labels and other visual attributes play an important role in graph perception.
- Graph interaction. User should have a possibility to interact with graph, zoom on it, select nodes and edges, add and remove elements, select subgraphs etc. This requirement is essential for the manipulation of tumor mapping graph.
- Good documentation. It allows to have a general overview of all possible features and to find solutions for emerging questions.

- Huge amount of nodes. Even if most of all modern graph libraries allow the visualization of thousands of elements, we consider important to include it into the list of requirements.

Despite the existence of different languages for data visualization, we will consider the two most popular ones: JavaScript and Python graph visualization libraries.

1.3.3. Python graph visualization libraries

NetworkX

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks (NetworkX, 2022). It has excellent documentation and huge number of features, such as graph generators, visual attributes and algorithms. However, NetworkX produces only static visualizations. NetworkX is one of the most popular graph libraries in Python with 11 000 stars on GitHub⁵.

Pyvis

A Pyvis network can be customized on a per node or per edge basis. Elements can have colors, sizes, labels and other metadata. The graph can be interactive, with all functionalities of dragging, hovering, and selection of the nodes and edges (Pyvis, 2018). Pyvis can be combined with NetworkX and it has 518 stars on GitHub⁶.

Bokeh

Bokeh is an open-source Python library for creating interactive visualizations for web browsers (Bokeh, 2022). This library is not focused on the node-edge diagrams, rather it allows the different formats of data visualization. As for other graph libraries, NetworkX can be integrated. Bokeh has more than 16 000 stars on GitHub⁷.

⁵ Available at: <https://github.com/networkx/networkx> Accessed in June 2022

⁶ Available at: <https://github.com/WestHealth/pyvis> Accessed in June 2022

⁷ Available at: <https://github.com/bokeh/bokeh> Accessed in June 2022

Dash Cytoscape

Dash Cytoscape is a graph visualization component in Python for creating easily customizable, high-performance, interactive, and web-based networks (Dash Cytoscape, 2019). It extends and renders Cytoscape.js, a JavaScript library of graph visualization. Dash Cytoscape library has good documentation with various examples. It has 446 stars on GitHub⁸.

1.3.4. JavaScript graph visualization libraries

Vis.js

Vis.js is dynamic browser-based visualization library which uses HTML canvas for rendering (Vis.js, 2022). The node-link diagram visualization represents the part of that library. Graph visualization supports custom shapes, styles, colors, sizes, and images. Vis.js also supports graph manipulation. It has 2 100 stars on GitHub⁹.

D3.js

D3.js is an open-source JavaScript library for manipulating documents based on data, using HTML, SVG, and CSS (Data-Driven Documents, 2022). The node-link diagram is one of the possible types of data visualization that can be performed with this library. D3.js uses pre-built JavaScript functions to select elements, create SVG objects, style them and add transitions and dynamic effects to them (Wang, Perez-Riverol, Hermjakob, & Vizcaino, 2015, p. 1356). However, for graph visualization, developers need to map data elements (nodes and edges) to visual elements (e.g., circles and lines), compute node positions, and calculate the start and the end positions of each edge according to its connected nodes (Han, Pan, Zhao, & Chen, 2021. p. 61). D3.js is one of the most popular libraries for data visualization with 102 000 stars on GitHub¹⁰.

⁸ Available at: <https://github.com/plotly/dash-cytoscape> Accessed in June 2022

⁹ Available at: <https://github.com/visjs/vis-network> Accessed in June 2022

¹⁰ Available at: <https://github.com/d3/d3> Accessed in June 2022

Sigma.js

Sigma.js is a JavaScript library for rendering and interacting with network graphs in the browser (Sigma.js, 2022). It uses Web Graphics Library (WebGL) to render graphs, which allows visualization of large graphs with thousands of nodes and edges. The graph data structure is managed in a separate library called Graphology. Sigma.js allows basic visualizations and interactions, however it is not easy to create custom ones. Another issue is the lack of documentation. Sigma.js has 10 000 stars on GitHub¹¹.

Cytoscape.js

Cytoscape.js is an open-source fully featured graph library for visualization and analysis written in JavaScript (Cytoscape.js, 2022). It is a highly optimized library that can be used for large and complex node-link diagrams. It uses the HTML canvas to render graphs. The Cytoscape.js architecture is composed of the core and the collection. The core represents the graph and is used to run layouts and animations, alter the view, and perform other operations on the graph as a whole (Franz, et al., 2016, p. 309). The collection represents the set of graph elements that can be used to filter, traverse or get data about those elements. The graph visualization can be highly customized. In addition, the library is fully extendable. Cytoscape.js includes different graph theory algorithms and has detailed documentation with comprehensive examples. There are 8 500 stars on GitHub¹² of Cytoscape.js library.

1.3.5. Graph libraries comparison

Comparative analysis of the graph libraries in the Appendix I shows that Cytoscape.js is the most appropriate library based on our requirements. Indeed, it is a popular open-source web-based library, that supports full customization and graph manipulation. It can also handle huge number of nodes and has a detailed documentation.

Another solution would be the Dash Cytoscape, which extends and renders Cytoscape.js with the Python language. It benefits from all functionalities of Cytoscape.js and offers the integration with the Dash components and NetworkX.

¹¹ Available at: <https://github.com/jacomyal/sigma.js> Accessed in June 2022

¹² Available at: <https://github.com/cytoscape/cytoscape.js> Accessed in June 2022

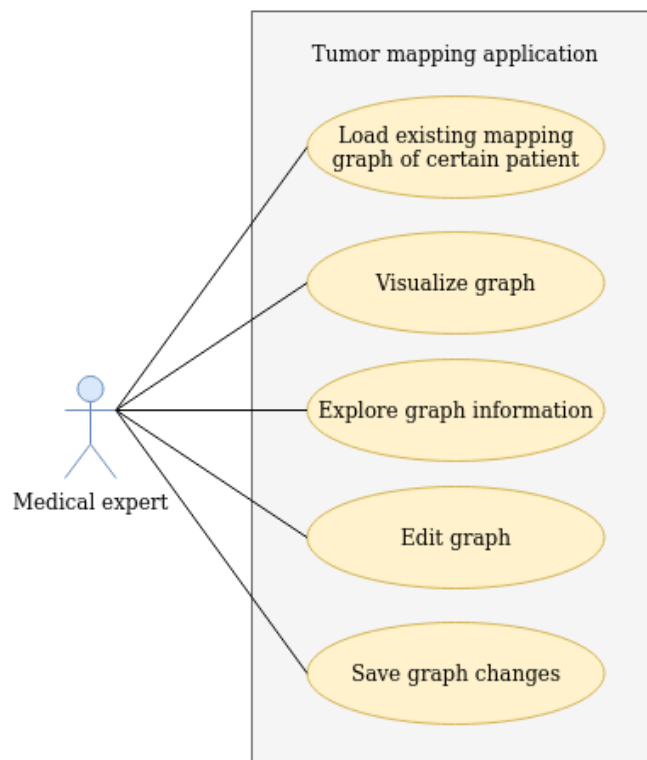
The choice between these two solutions can be done easily since they represent the same library but for different programming languages. Thereby, the selection between Python Cytoscape and Javascript Cytoscape should be based on the chosen application architecture.

In the next two chapters, we will define the use cases that can be performed within the application as well as describe an application architecture.

1.4. Use cases definition

In order to identify and clarify the requirements for our application from the user-side perspective, we need to determine the use cases. According to our goal, we define the following use cases that a user can perform within tumor mapping application (Figure 4):

Figure 4: Use Cases of tumor mapping application



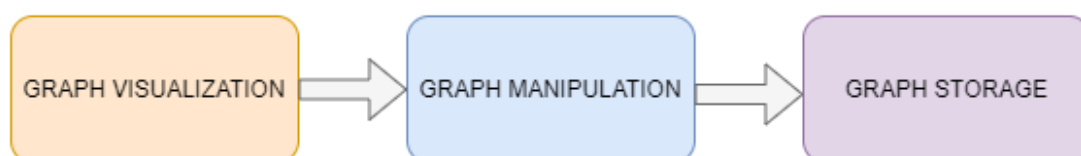
Source: author's source

- Load existing mapping graph of certain patient. The user accesses the application, enters the patient identification number, and receives the patient graph of tumor mappings.

- Visualize graph. The user observes the graph, the benign and suspicious ROIs and the connections between them. The graph representation gives a clear overview of ROIs volumes, ROI types as well as probability of correspondences across imaging timepoints. The user can zoom in the graph for better perception.
- Explore graph information. The user can see the information concerning selected node or edge. At the same time, the user can perform filtering of the graph based on the threshold value. Threshold value represents the automatically defined overlap between two ROIs. The user can also overview different subgraphs of all related ROIs.
- Edit graph. The user can edit the graph by removing or creating edges as well as changing the information of existing nodes and edges.
- Save changes of graph. After completing the process of the manual overview of the graph, user can save the result of the work and retrieve it again later with the patient's identification number.

Based on the described use cases, we can define three main parts of tumor mapping application: graph visualization, graph manipulation and graph storage. We consider the term “graph visualization” as a common notion for the graph display, which signifies the visual perception of the graph, and the graph interaction, which stands for actions that can be performed on graph, such as clicking, hovering and zooming the elements of the graph. Graph manipulation represents the actions that provoke changes in the state of existing graph, such as removing and creating of edges, or changing node information. Graph storage is responsible for storing and retrieving the graph of certain patient from the database. The flow of the development of these parts is presented in the Figure 5.

Figure 5: Development phases of tumor mapping application based on use cases



Source: author's source

In the next chapter, we will define the architecture of the application, which will allow us to implement the described use cases.

1.5. Application architecture

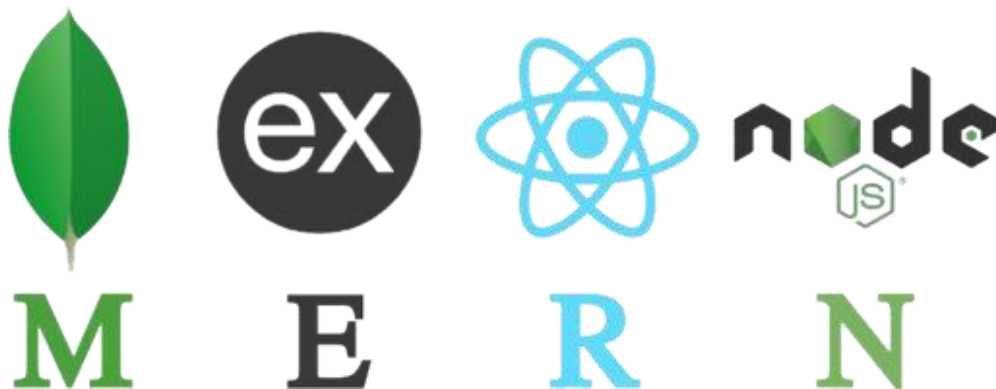
In order to provide an optimal solution for our application, we will use the traditional three-tier architecture. Such architecture consists of the presentation tier, logic tier, and data tier and allows the application to be flexible and reusable. Indeed, our application should have web-based interface to display the graph, store permanently the graph into the database in case of changes and have a connection between these two systems in form of Application Programming Interface (API).

It exists a lot of possible solutions for the implementation of such architecture based on different programming languages and different approaches. One of the approaches is a server-side rendering (SSR). With this technique, the result of the request from the webpage is rendered on the server and the output of that rendering is sent back to the browser. Another approach is the client-side rendering (CSR). In this case, the result of the request is rendered directly in the browser with the help of JavaScript.

Both approaches have their strengths and weaknesses, depending on the goal of the utilization. Fast loading of the initial page, availability for research engines indexing and high security are the main advantages of the SSR method. At the same time, it demands frequent server requests and, as a consequence, full page reloads, which influence on the overall page rendering. The CSR application renders the webpages rapidly, but the loading time of the initial page is increased (Thakkar, 2020, p.93). Generally, the server-side rendering is used more frequently for static sites, whereas the client-side rendering is used for the web applications that demand a lot of interactions.

Since our study is focused on the web-based visualization of the graphs, we will implement the client-side rendering approach, because it allows rich and fast interactions within the web application. For that purpose, we will use the stack of technologies known as MERN.

MERN stands for MongoDB, Express, React, Node and represents the JavaScript stack that is used for the creation of full-stack web applications (Figure 6).

Figure 6: MERN stack components

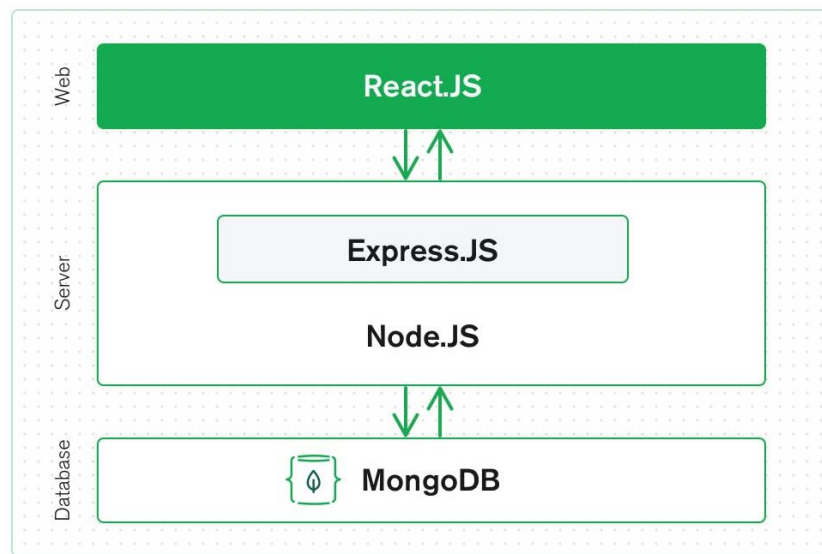
Source: [wikimedia.org](https://commons.wikimedia.org/wiki/File:MERN-logo.png) ¹³

MERN stack consists of four key technologies:

- MongoDB. Represents the document-oriented, NoSQL database to store the application data.
- Node.JS. Represents a cross-platform JavaScript runtime environment that allows to build diverse tools, servers and applications (Koroliova, 2018, p.8).
- Express.JS. Represents the Node.js web framework that is used to build the backend of the web application, using NodeJS functions and structures.
- React.JS. Represents the Javascript library for creating user interface of the single page web applications.

MERN is a full-stack solution, that follow the traditional three-tier architectural pattern, including the front-end display tier (React.js), application tier (Express.js and Node.js), and database tier (MongoDB). MERN is entirely using JavaScript and JSON (MongoDB, 2022). Figure 7 represents the three-tier architecture of the MERN.

¹³ Available at: <https://commons.wikimedia.org/wiki/File:MERN-logo.png> Accessed in June 2022

Figure 7: MERN stack three-tier architecture

Source : mongodb.com¹⁴

The main advantages of using MERN stack are the possibility of full-stack development with only one programming language (Javascript), easy learning, strong community support, flexibility and popularity (Kapoor, 2021). That is why we consider this set of technologies as an optimal solution for our application.

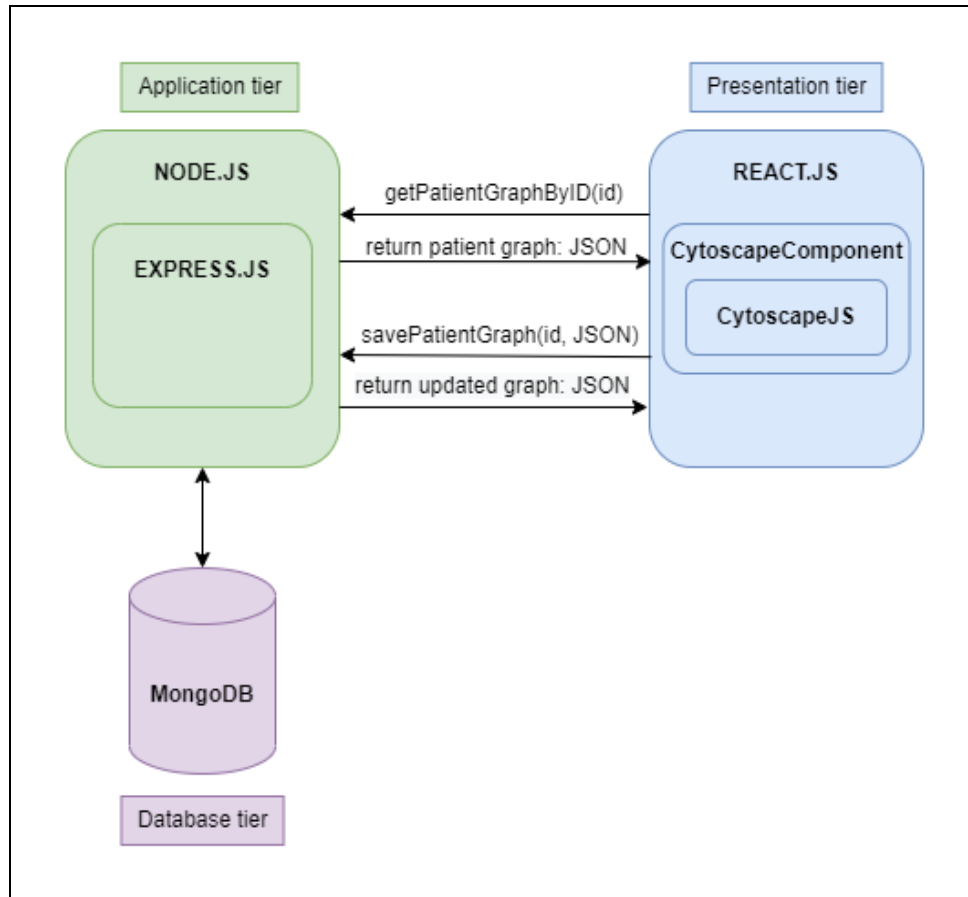
Based on that stack of technologies, our application architecture will be the following:

- Presentation tier. With React.JS, we will create a user interface which will be used by an expert for the graph visualization and graph manipulation. The graph visualization library Cytoscape.js will represent our final choice of the library, as it is the most appropriate solution for the MERN stack. This library will be implemented as a React component with help of React-CytoscapeJS wrapper.
- Application tier. We will use Node.JS and Express.JS to create a RESTful web API for the data exchange between the React.JS and MongoDB. The RESTful API uses HTTP requests to access and use data (Bojinov, 2016).
- Database tier. The graphs of patients will be stored in the MongoDB database in the JSON format.

¹⁴ Available at: <https://www.mongodb.com/mern-stack> Accessed in June 2022

Figure 8 shows the relations between three systems of the tumor mapping application.

Figure 8: Three-tier architecture of tumor mapping application



Source: author's source

The client system, which is the presentation tier, communicates with the application tier in two cases. Firstly, to request the tumor mapping graph of certain patient, based on the identifier (ID) of that patient. It uses the GET method of the RESTful API. Secondly, to send the updated graph in reason to store it into the database. In this case, the PUT method of the RESTful API is used.

In the METHODOLOGY section, we considered what a graph is, which file formats of graph data exist and what is the most appropriate graph visualization library for our study. We also defined use cases and application architecture. In the next section, we will describe the results of our study.

2. RESULTS

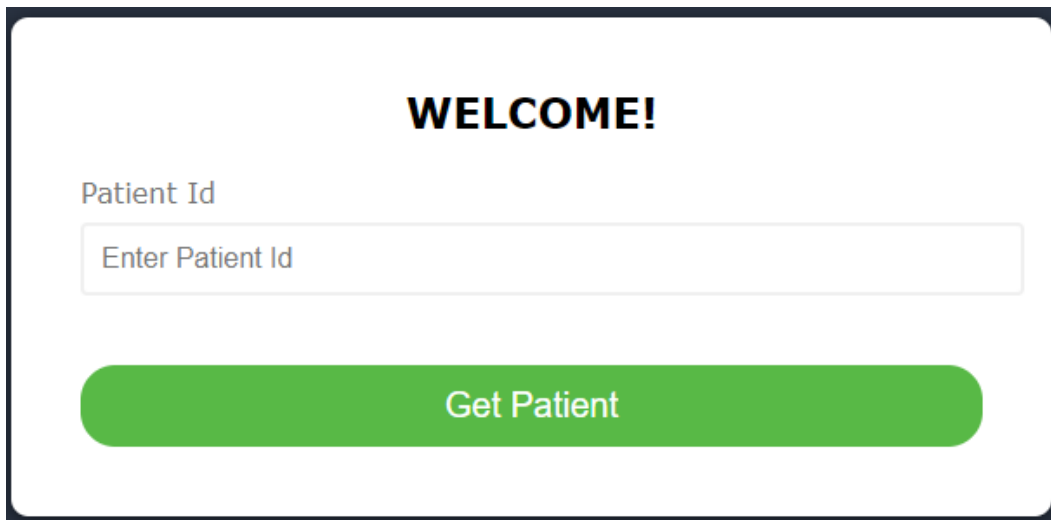
This section is devoted to describing the results of our study, which are based on a previously defined methodology. To begin with, we will consider the general overview of the application. Then, we will overview provided data of the graph. After, we will describe how the graph can be visualized and changed within our web-based application. Finally, we will review the process of graph storage.

2.1. General overview

The application is composed of two pages: the home page, where the user needs to enter the patient identifier to retrieve the tumor mapping graph, and the main page, where the user works with the patient graph. We refer to a radiologist or a researcher when we speak about the user of the application.

The home page has a simple interface with one input form (Figure 9).

Figure 9: Input form of the home page



The screenshot shows a web interface for a home page. At the top, the word "WELCOME!" is displayed in bold, black, uppercase letters. Below this, the text "Patient Id" is shown in a smaller, blue font. Underneath, there is a white rectangular input field with a light gray border and the placeholder text "Enter Patient Id" in a light gray font. At the bottom of the form, there is a large, rounded green button with the text "Get Patient" in white, bold, uppercase letters.

Source: author's source

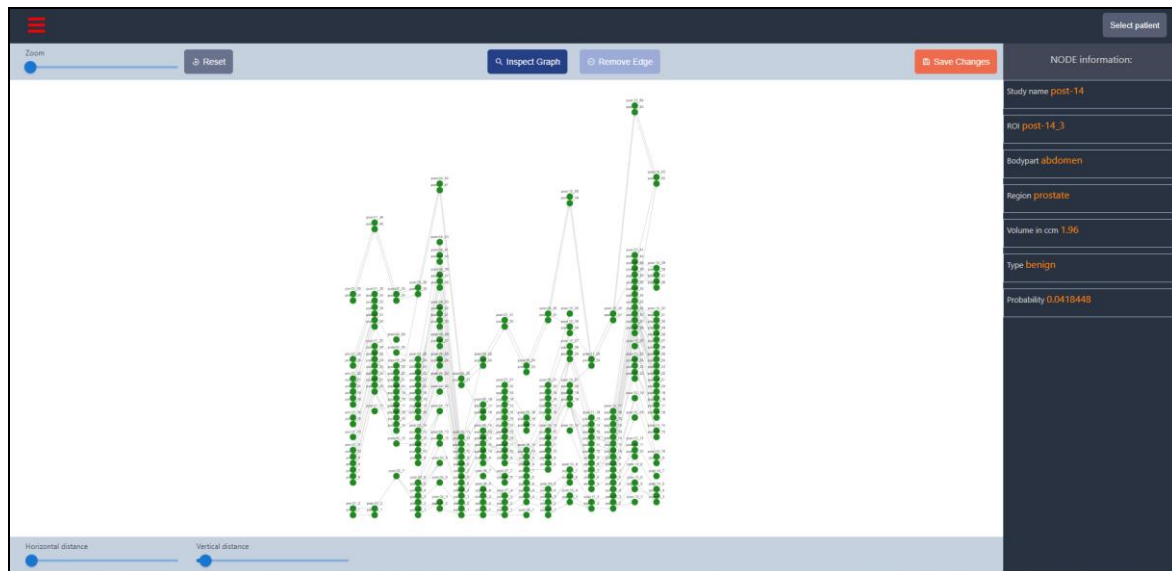
The input field accept only numbers. When the user clicks on the “Get Patient” button, the application redirects to the main page.

If the patient ID does not exist in the database, an alert window is shown, and the application redirects back to the home page.

If the patient ID exists in the database, the tumor mapping graph is displayed on the main page. From this moment, the user can work with the graph.

The general overview of the main page can be seen in the Figure 10.

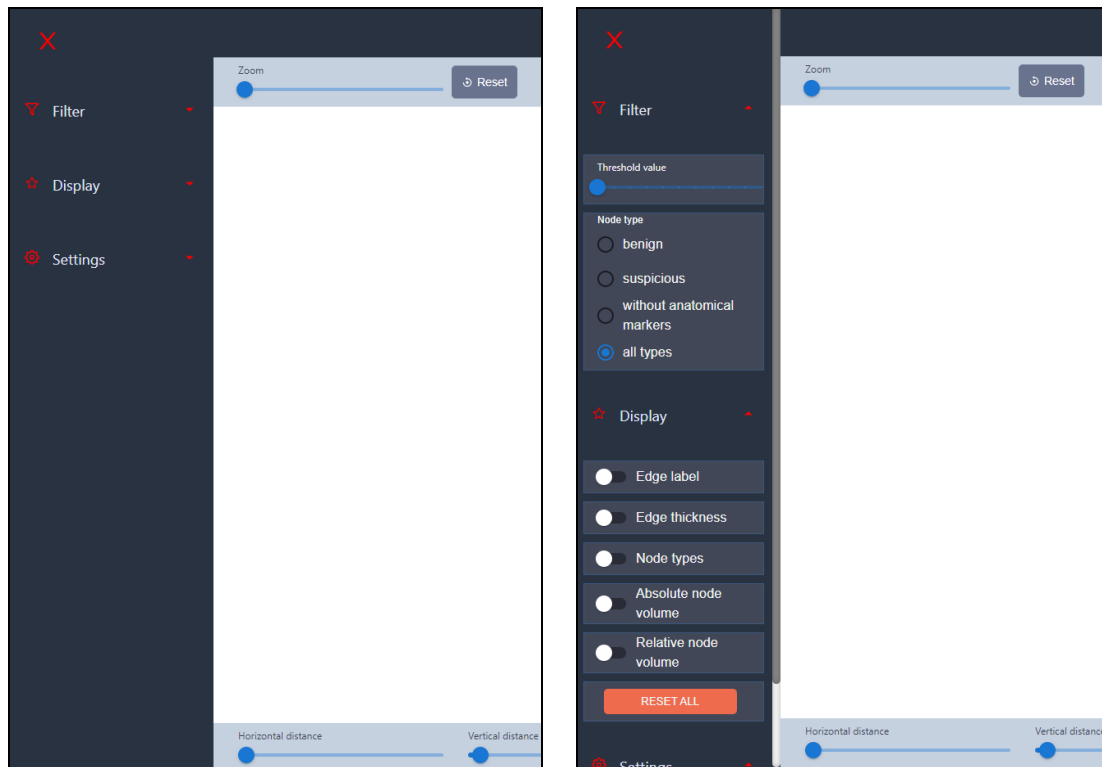
Figure 10: Main page of the application



Source: author's source

In general, we can distinguish the following sections of the main page:

- Navigation bar. Located on the top of the page, it has two buttons: "Select patient" and "Open sidebar". The "Select patient" button redirects the user to the home page, so he/she can choose another patient. "Open sidebar" allows to open a hidden sidebar.
- Graph area. Represents the container, where the graph is displayed.
- Graph toolbars. The blue-grey areas that contain sliders and buttons to be used on the graph.
- Right panel. Represents the region, where the node or edge information is displayed.
- Sidebar. Contains the visual settings tools. Figure 11 shows different views of the sidebar, depending on whether the submenus are open or not.

Figure 11: Open sidebar (in the left) and open submenus of sidebar (in the right)

Source: author's source

The functionality of the application will be covered in the following chapters.

2.2. Graph data representation

In this chapter we will consider the graph data that is provided to our application in order to construct the graph.

The nodes and the edges are referred as the elements in the Cytoscape.js library. To construct a graph, the Cytoscape.js defines these minimal requirements for the elements:

- All elements should have an ID. It will be assigned automatically if it is undefined.
- Edge element should have “source” and “target” properties, which allow Cytoscape.js to identify the element as an edge. Source property defines from which node the edge comes and target property refers to the node where the edge goes.
- The elements data should be in JSON format.

The data related with the elements can be defined arbitrarily. The Figure 12 shows the example of the node element of the tumor mapping graph.

Figure 12: Node structure of tumor mapping graph

```
{
  "data": {
    "label_id": 5,
    "pos": [
      0,
      5
    ],
    "study_name": "pre-01",
    "roi_id": 4,
    "pars_bodypart": "abdomen",
    "pars_region": "rectum",
    "vol_ccm": 1.97,
    "type": "benign",
    "suspicious_probability": 0.0663896,
    "id": "pre-01_5",
    "value": "pre-01_5",
    "name": "pre-01_5"
  }
}
```

Source: author's source

The node element has the following attributes:

- **id:** represents the node identifier and is composed of study_name, label_id, and the underscore between them.
- **study_name:** indicates a temporal position relative to the dates of imaging examinations, for example, pre-01, post-01, post-02.
- **label_id, roi_id:** indicates ID of the ROI at the respective timepoint. Label_id starts from 1, roi_id starts from 0.
- **pars_region, pars_bodypart:** anatomical location of ROI according to the segmentation software.
- **vol_ccm:** volume of the ROI in cubic centimeters.

- **suspicious_probability**: probability that ROI represents a malignant lesion. By default, the probability > 0.5 is interpreted as the ROI being a malignant lesion.
- **type**: indicates a binarized malignancy status based on “suspicious probability”. It can be rather benign or malignant. If a ROI represents the anatomical marker, the type is undefined.
- **pos**: indicates the X and Y node positions. X position is based on the study_name timepoint. Y position is based on the label_id.
- **value, name**: hospital identifiers.

The example of the edge element of the tumor mapping graph is shown in the Figure 13.

Figure 13: Edge structure of tumor mapping graph

```
{
  "data": {
    "intersection_mean": 0.6103987807418616,
    "intersection_std": 0.07109132496572766,
    "count": 10,
    "source": "pre-01_5",
    "target": "post-01_3"
  }
}
```

Source: author's source

The edge element has the following attributes:

- **source**: indicates from which node the edge comes.
- **target**: indicates to which node the edge goes.
- **count**: indicates number of registrations attempts that lesion pair was involved in. The regions that contain multiple densely clustered ROIs are co-registered several times. For each co-registered pair of image sub-regions, spatial overlap is computed between all pairs of delineated ROIs present in these sub-regions. Therefore, the overlap between any pair of ROIs may be assessed multiple times. The count attribute indicates the number of assessments (Abler, 2022, p.4)

- **intersection_mean:** indicates the average intersection of connected lesions across multiple overlap assessments.
- **intersection_std:** standard deviation of intersections across multiple overlap assessments.

The provided data are fully anonymized. The hospital identifiers have been replaced by another set of identifiers, the patient names have been removed, the dates of imaging examinations have been replaced by special identifiers relative to the start of treatment (pre-01, post-01, post-02 etc.).

2.3. Graph visualization

This chapter is dedicated to a detailed description of the process of the graph displaying and interaction.

2.3.1. Graph layout

The first step of graph visualization is the selection of an appropriate layout. The function of a layout is to set the positions of the nodes in the graph (Cytoscape.js, 2022). Despite the existence of a huge number of different layouts, we need a specific one, that allows to arrange the nodes in the time-oriented sequence. Cytoscape.js library provides a lot of different layouts, but not the time-oriented one. For that reason, we use the custom layout, where the positions of nodes need to be set explicitly.

The positions of the nodes, specified in the node data, have unique values, since they are based on the ROI identifier and timepoints. However, the sequential allocation of identifiers causes too close location of nodes to each other. For example, if the first node has a position of [0, 5] and the second one has a position of [0, 6], the node images will be overlapped.

Another issue is that the node positioning starts in the top and left of the canvas. Therefore, a node with the position [0, 0] will be displayed in the upper left corner and a node with the coordinates [0, 10] will be displayed under the first node. However, for better visual perception of the timepoints, we need to use a standard representation of the time and value axes, where the positioning starts in the bottom left corner and grows up when the ROI id increases.

To overcome these problems, we define a small algorithm to change the node position and to invert the graph in vertical axis (Figure 14).

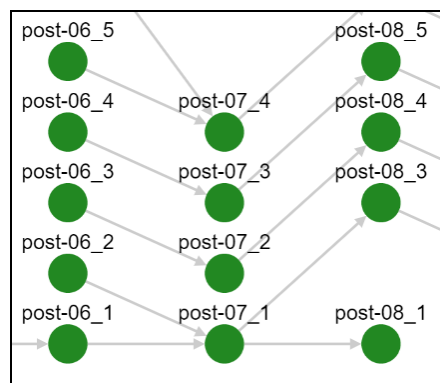
Figure 14: Node position changing algorithm

```
transform: function (node, position) {
  const pos = node._private.data.pos;
  node.position({ x: pos[0] * deltaX, y: - pos[1] * deltaY });
  position.x = node.position("x");
  position.y = node.position("y");
  return position;
},
```

Source: author's source

First, we introduce two global variables, `deltaX` and `deltaY`, that represent the distance between nodes in X and Y axes respectively. The position of each node is multiplied by these deltas. Properly selected deltas allow to overcome the problem of overlapping as it increases the distance between nodes. To inverse the graph, we use the negative value of Y position, so the graph is displayed in the desired way. Such algorithm doesn't change the nodes position data, but only the graphical location of nodes, so it is always possible to retrieve the original positioning.

Figure 15: Time-oriented graph layout

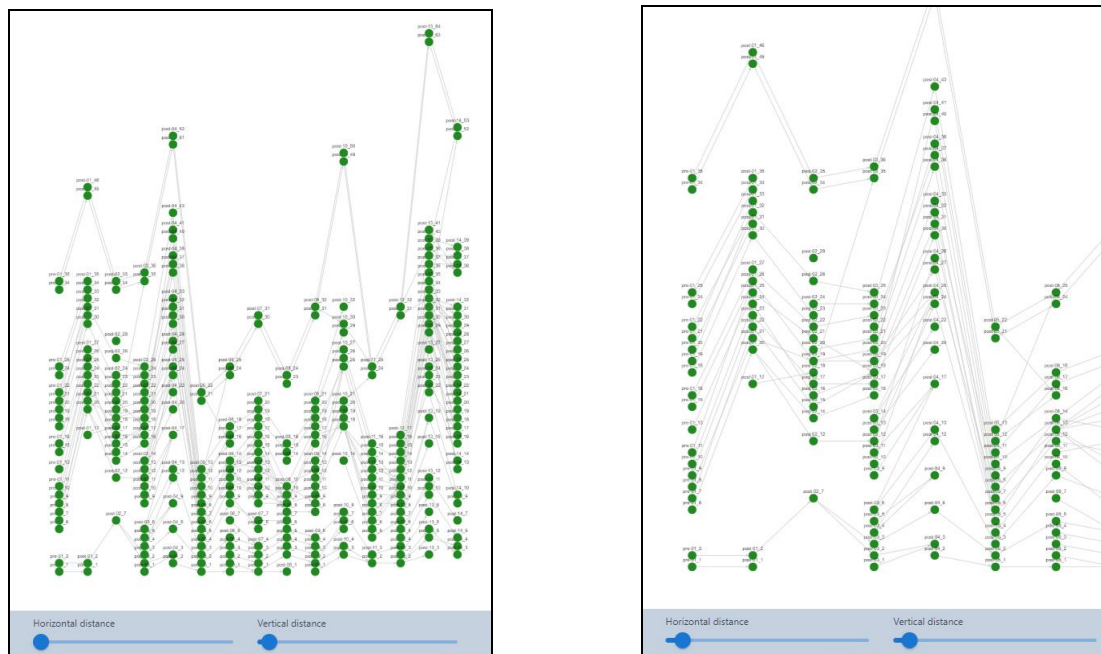


Source: author's source

The Figure 15 shows the graph in time-oriented layout. In this example, we can observe three timepoints `post-06`, `post-07` and `post-08` that represent three dates of imaging examinations. Each timepoint has a set of ROIs that were automatically detected. In other words, one column represents one timepoint and the nodes in this column represent all detected ROI at that timepoint.

To configure the graph visualization, the user can change the values of delta variables. The Figure 16 shows how the graph visualization can be changed based on the node distances. The left image represents the basic configuration of the nodes distance with delta values set by their default (20 for the deltaX, and 30 for the deltaY). The right image shows the increased horizontal and vertical distances between nodes by increasing of these deltas.

Figure 16: Basic node distance (in the left) and increased distance (in the right)



Source: author's source

To increase or decrease those values, the user needs to use the distance sliders on the bottom tools panel. The “Horizontal distance” slider manages the X position of the nodes and the “Vertical distance” slider manages the Y position. Such distance control functionality facilitates the graph perception, since it allows the user to easily configure comfortable distances between nodes, which can be especially useful in case of nodes size changes.

Another configuration that can be done to facilitate the graph perception, is the zooming of the graph. For that, the user can use the scroll wheel of the mouse or use the zoom slider on the top tools panel. In addition, the graph can be easily moved on the canvas. Distance between nodes, position of the graph on the canvas and graph zoom can be reset to their basic values with the button “Reset”. The basic view of the graph will always fit the canvas, regardless of the graph size.

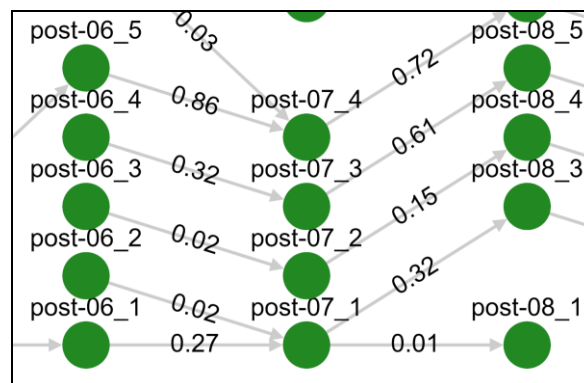
2.3.2. Visual features

In this chapter, the features of graph displaying will be discussed. By display features we understand all visual changings without structural modification of the original graph.

1. Edge label

Edge labeling allows to see the average intersection of a pair of nodes. The values are taken from zero to one, where one means 100% of intersection. The Figure 17 shows the enabled “edge label” option.

Figure 17: Example of enabled “edge label” feature

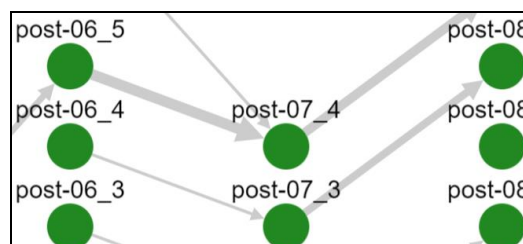


Source: author's source

2. Edge thickness

Edge thickness offers an alternative way to show the average intersection of nodes. The width of the edge is calculated based on which cluster the intersection mean belongs to. We define ten clusters, i.e., 0...0.1, 0.1...0.2, 0.2...0.3 etc. The higher the “intersection mean” value, the thicker the line of the edge (Figure 18).

Figure 18: Example of enabled “edge thickness” feature

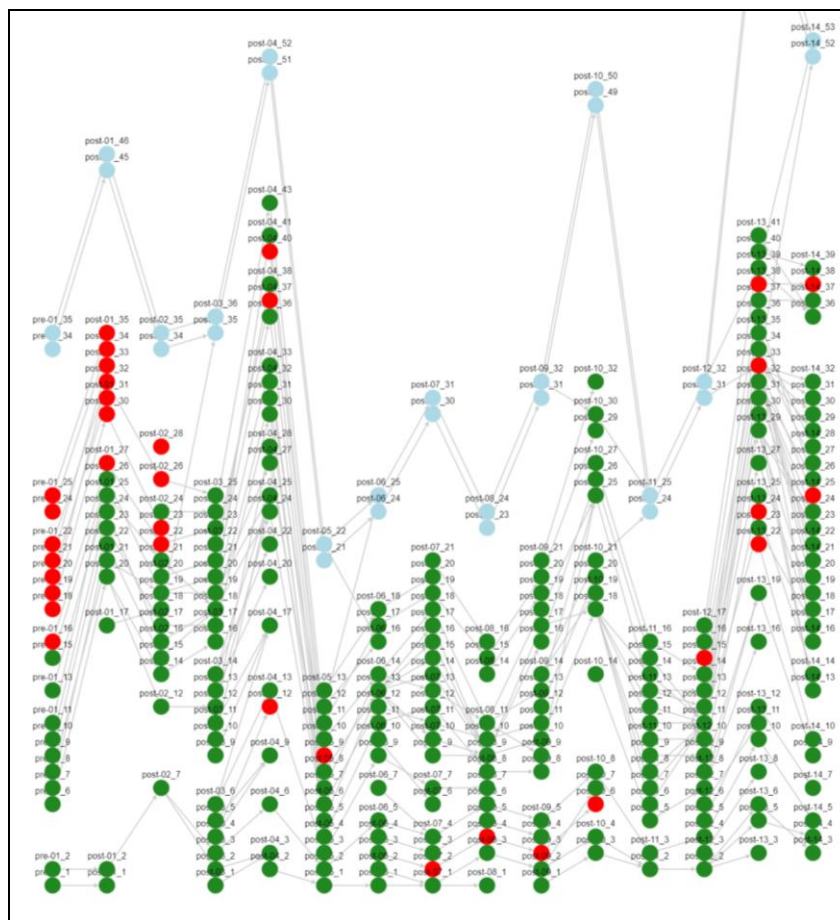


Source: author's source

3. Node types

Node type represents the defined type from the node data, which can be “benign”, “suspicious” or “undefined”. The last type appears when the node is considered as the anatomical marker. To distinguish the node types, different colors are used. The red color is used for the suspicious nodes, the green one for the nodes considered as benign and the blue one for the anatomical markers. It allows to easily distinguish nodes (Figure 19).

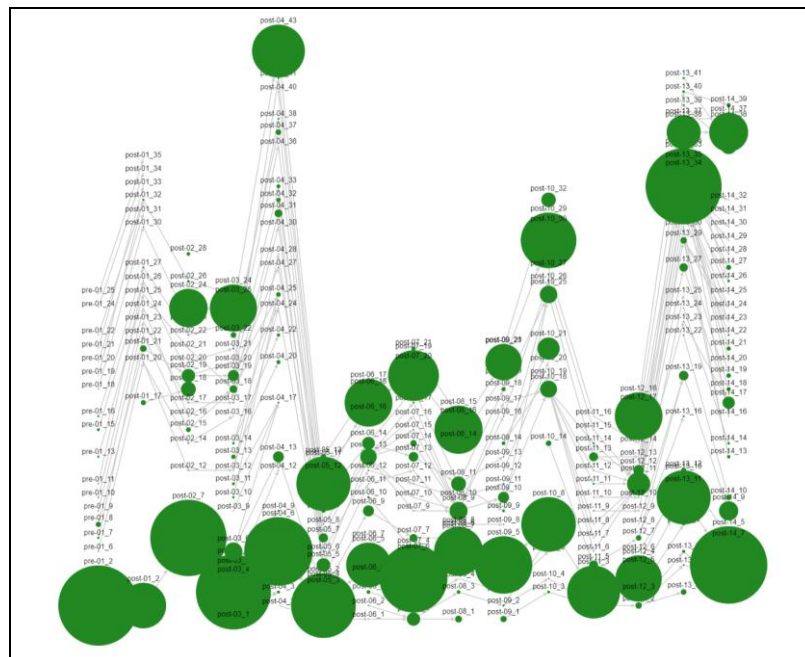
Figure 19: Example of enabled “node types” feature



Source: author's source

4. Absolute node volume

Absolute node volume represents the size of the node based on the node data property “vol_ccm”. It allows the user to rapidly perceive the volume difference between the ROIs (Figure 20).

Figure 20: Example of enabled “absolute node volume” feature

Source: author's source

5. Relative node volume

Although the “absolute node volume” option gives the complete representation of the volumes of ROIs, it can be sometimes hard to work with the graph due to huge differences between node volumes, where some ROIs are several times bigger than the others. For that reason, we introduce the “relative node volume” option. Figure 21 shows the computing algorithm of relative node volumes.

Figure 21: Relative node volume algorithm

```
const maxVolCcm = Math.max(...nodes.map((x) => x.data.vol_ccm));
const minVolCcm = Math.min(...nodes.map((x) => x.data.vol_ccm));
const nb_clusters = 5;
const step = (maxVolCcm - minVolCcm) / nb_clusters;

for (let i = 1; i <= nb_clusters; i++) {
  cy.style()
    .selector(`node[vol_ccm <= ${i * step}]`)
    .selector(`node[vol_ccm > ${i * step}]`)

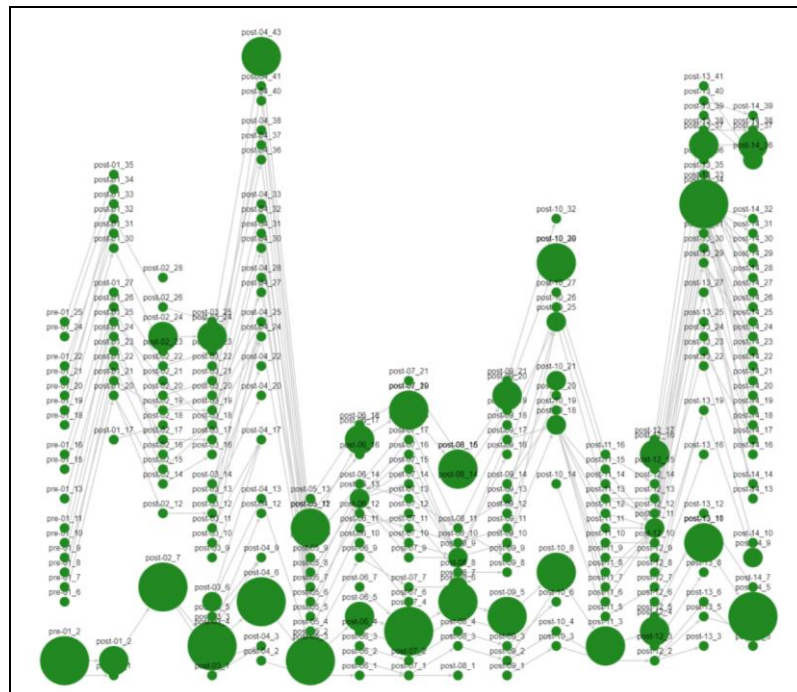
    .style({
      width: `${(200 * i) / 10}`,
      height: `${(200 * i) / 10}`,
    })
    .update();
}
```

Source: author's source

To calculate the relative node volume, first we compute the maximum and the minimum nodes volumes, based on the “vol_ccm” property. After that, we define the number of clusters into which we want to divide all the nodes. Then, we compute the “step” to define those clusters. And finally, we allocate each node to one of cluster and set its style.

For example, if the minimum volume value of all nodes is 10, and the maximum is 120, the step of the cluster will be equal to $(120 - 10) / 5 = 22$. So all nodes, that have a volume between 10 and 22 will be members of the first cluster, all nodes with the volumes between 23 and 44 will be assigned to the second one etc.

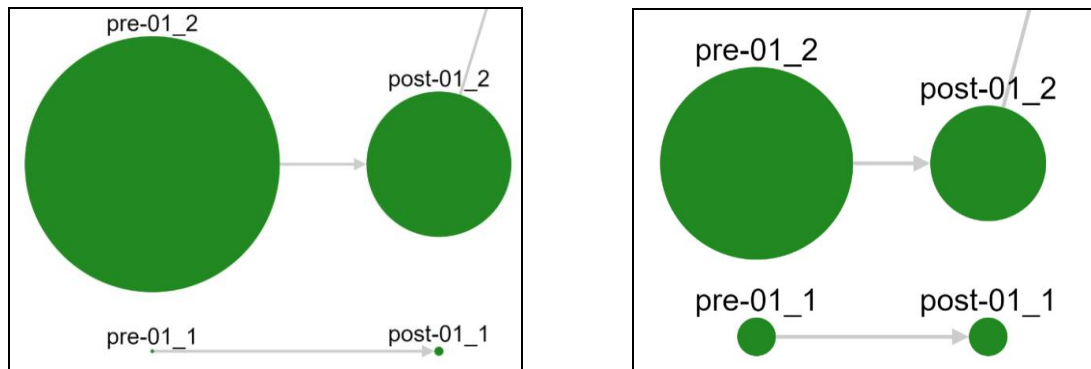
Figure 22: Example of enabled “relative node volume” feature



Source: author's source

Figure 22 represents the general overview of “relative node volume” option applied on the graph.

Figure 23: Comparison between absolute (in the left) and relative (in the right) node volume options



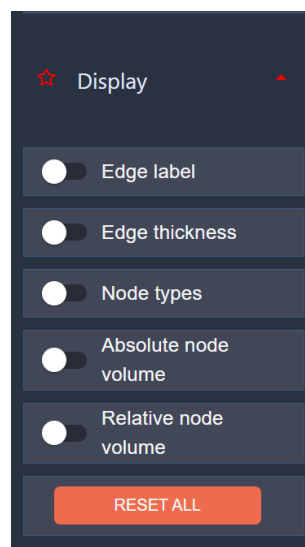
Source: author's source

The Figure 23 shows the comparison between absolute and relative node volume options. The ROI “pre-01_2” has the volume of 178.2 ccm, while the ROI “pre-01_1” has the volume of 2.64 ccm. With the relative node volume option, it is still possible to see the proportions, but the graph looks more lightweight.

6. Configuration panel

All visual features can be configured on the “Display” menu of the sidebar (Figure 24). It is possible to combine all features, except absolute and relative node volume options, which are opposites.

Figure 24: Configuration panel of visual features



Source: author's source

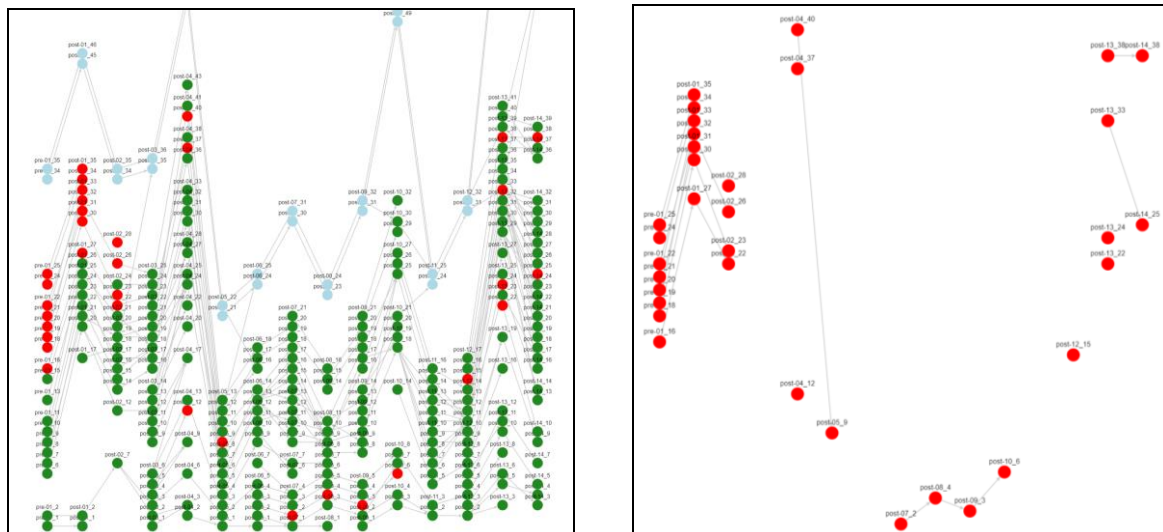
2.3.3. Filtering features

In this chapter, the filtering that can be applied on the graph will be discussed. The filtering allows the changing of the graph representation based on specific parameter.

1. Node type filtering

Node type filtering is used to display the nodes based on the node type, which can be “benign”, “suspicious” or “undefined”. Such filtering allows the user to work with filtered graph, for example only with those nodes which are defined as “suspicious” (Figure 25).

Figure 25: Node type filtering: basic graph (in the left) and filtered by suspicious node type graph (in the right)



Source: author's source

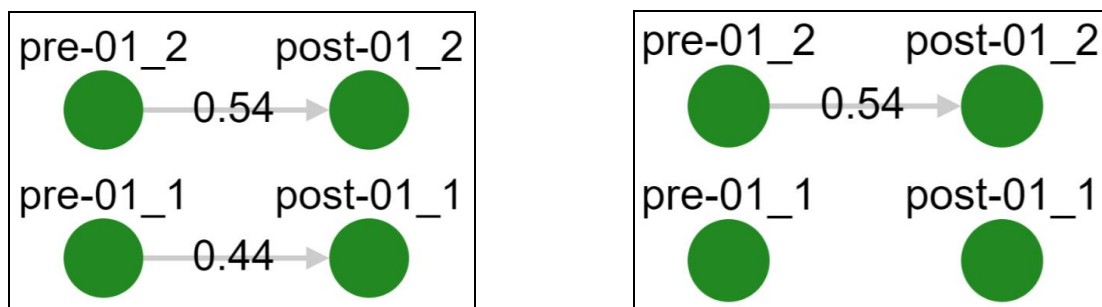
The application has four options for node type filtering:

- Benign. Display all nodes that have a type “benign”.
- Suspicious. Display all nodes that have a type “suspicious”.
- Without anatomical markers. Display benign and suspicious nodes.
- All types. Display all nodes. This option is set by default.

2. Threshold filtering

Threshold filtering allows to filter edges based on the “intersection mean” value. The ROIs, connected by the edges with high average intersection value, are more important to the expert evaluation process, compared to average intersection values close to zero. Applied threshold filter removes all edges with the “intersection mean” smaller than the threshold value (Figure 26).

Figure 26: Threshold filtering: threshold filter set to 0 (in the left) and to 0.5 (in the right)

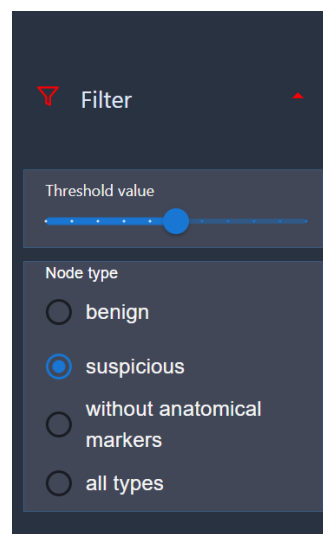


Source: author's source

3. Configuration panel

The node type filter and the threshold value can be configured on the “Filter” menu of the sidebar. It is possible to combine both filters as well as all visual features (Figure 27).

Figure 27: Configuration panel of filtering



Source: author's source

2.3.4. Graph interaction

In this chapter, the interaction with the graph will be described. We intentionally distinguish two notions: graph interaction and graph manipulation to separate those two processes. By graph interaction we understand actions that can be performed on graph, such as clicking, hovering and displaying of subgraph. Graph manipulation represents the actions that change the initial state of the graph, for example removing an edge. Graph manipulation will be covered in the next chapter.

1. Hovering over element

In order to observe the information about an element, the user needs to hover the cursor over the element of interest. The information about the element will be shown on the right panel (Figure 28).

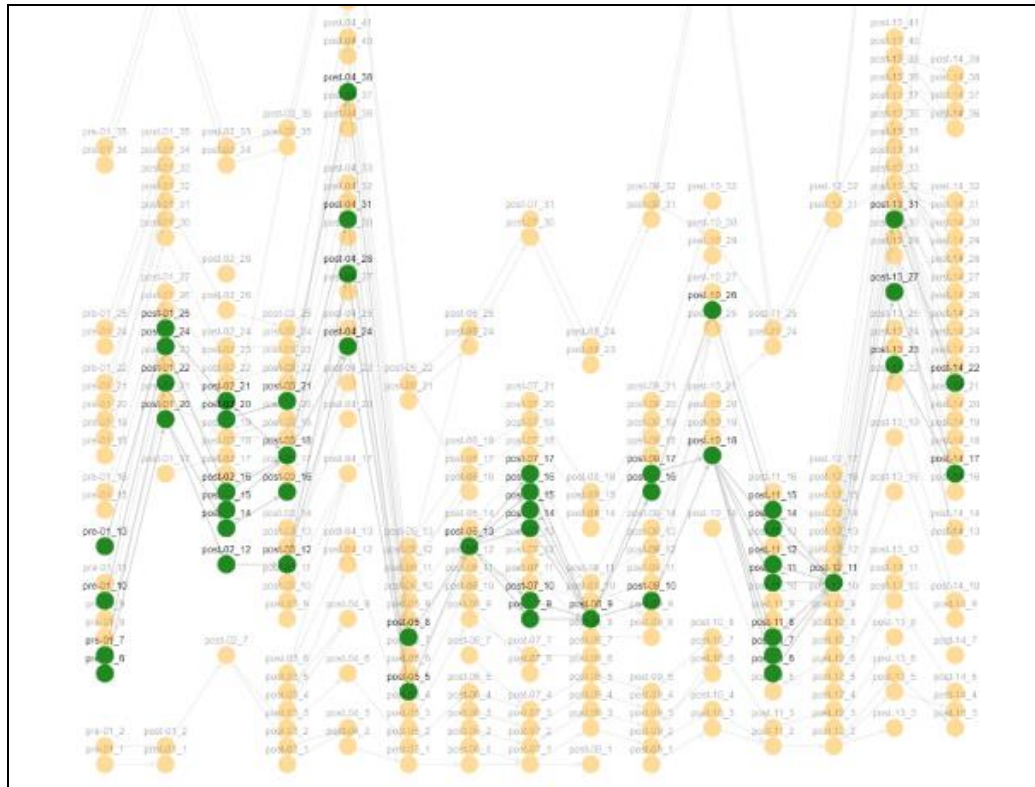
Figure 28: Examples of node and edge information

NODE information:	EDGE information:
Study name post-14	From post-13_6
ROI post-14_3	To post-14_3
Bodypart abdomen	Intersection mean 0.092907
Region prostate	Standard deviation 0.014843
Volume in ccm 1.96	
Type benign	
Probability 0.0418448	

Source: author's source

At the same time, hovering over the node or the edge allows the user to see all related elements. It helps to easily observe the relations between nodes. The Figure 29 shows how the subgraph of all related ROIs is highlighted when hovering over one of its elements.

Figure 29: Graphical representation of all related ROIs on element hovering



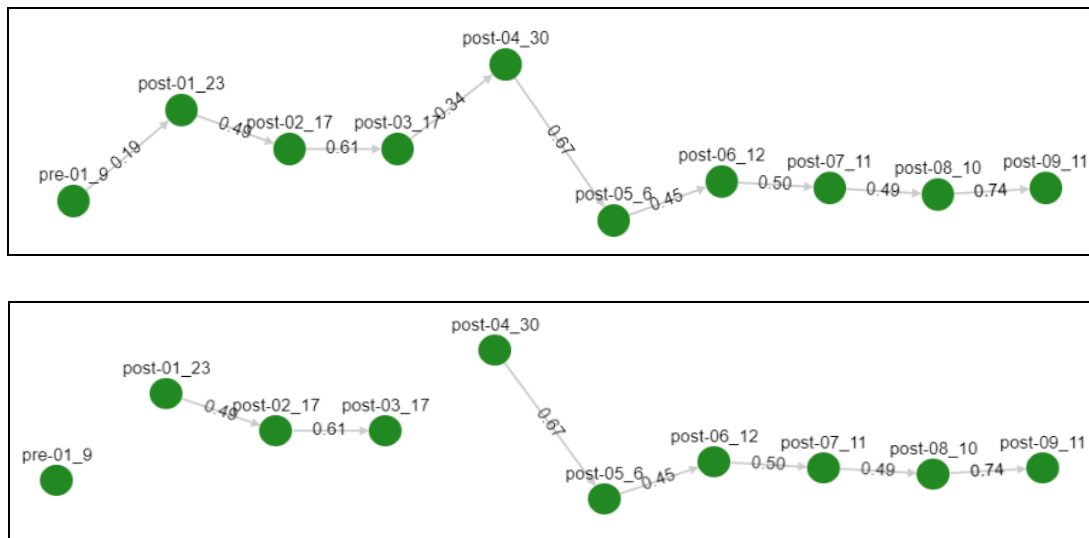
Source: author's source

2. Clicking on element

When the user clicks on a node or on an edge, the subgraph of all connected nodes is selected, and the rest of graph is removed from the view. Now that subgraph becomes the main graph. The user can apply filters or change the visual settings of that subgraph. The threshold filtering, applied on that subgraph, can divide it into smaller subgraphs.

When the first subgraph is selected, the “Return” button appears. From that moment, the user can navigate back to the original graph. All subgraphs are added into a history list, which allows to store the sequence of the selected subgraphs.

Figure 30: Basic subgraph (top image) and three new subgraphs after threshold value filtering (bottom image)



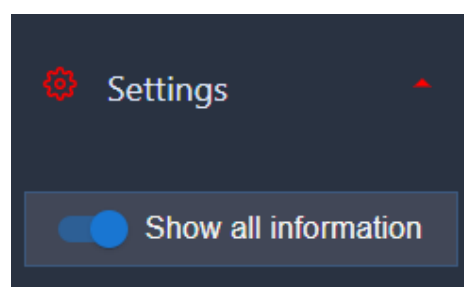
Source: author's source

The Figure 30 shows the separation of one subgraph into three new subgraphs when the threshold values is set to 0.4. At that moment, it is also possible to inspect any of those subgraphs.

3. Settings

In the basic configuration, when the user hovers over an element, the information, that is considered the most relevant, is displayed. However, in some cases, the user wants to see the full information about the element. For example, the researcher might want to see the hospital ID of the ROI. For that reason, we provide the possibility to switch between those two modes: display the most important information or display the full information. It can be configured in the “Settings” menu of the sidebar (Figure 31).

Figure 31: Settings menu



Source: author's source

2.4. Graph manipulation

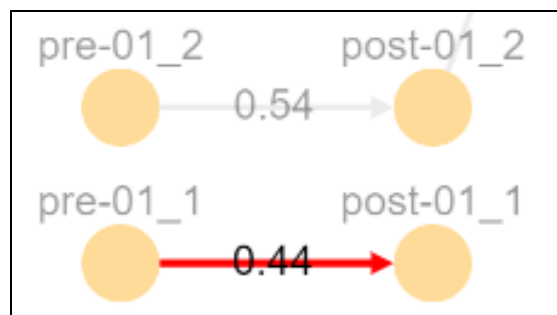
This chapter describes the manipulation of the graph. As discussed before, we define the graph manipulation as the actions that change the state of the existing graph. Some examples of those actions can be removing and creating edges or changing the node information.

The application provides two modes of interaction with the graph: “Inspect Graph” and “Remove Edge”. The “Inspect Graph” mode allows the user to hover over an element to see the detailed information about this element, as well as to select a subgraph on clicking on the element. The “Remove Edge” mode changes the “on hover” and “on click” actions.

1. Hovering over edge

While the “Remove Edge” mode is activated, the user can hover over the edges to select those to be removed. While hovering over an edge, it will be highlighted in red and the other elements will be put in the background (Figure 32).

Figure 32: Example of hovering over edge



Source: author's source

2. Clicking on edge

Any edge can be removed from the basic graph. To remove an edge, the user needs to click on it. A confirmation window will appear and after confirmation, the edge will be removed from the graph. The “remove” action can be reverted by refreshing the application web page.

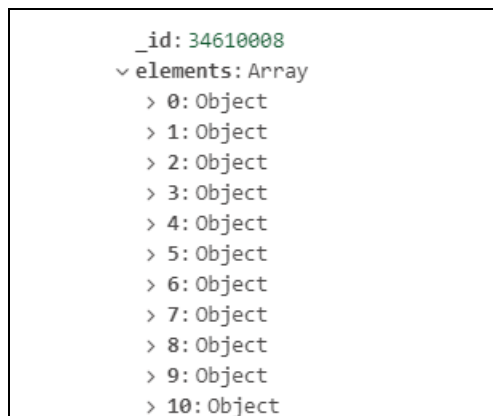
2.5. Graph storage

In this chapter, the storage of the graph data will be described. It will be considered through a three-tier architecture: the database, the backend, and the frontend.

1. Database

We use MongoDB to store the graph data. The database has one collection “patients”, where all graphs are stored. Each element of that collection consists of the patient ID and the array of the elements. Cytoscape.js can distinguish whether an element is a node or an edge, so there is no need to store this information (Figure 33).

Figure 33: Example of one element of the Patients collection in MongoDB



```
_id: 34610008
  elements: Array
    > 0: Object
    > 1: Object
    > 2: Object
    > 3: Object
    > 4: Object
    > 5: Object
    > 6: Object
    > 7: Object
    > 8: Object
    > 9: Object
    > 10: Object
```

Source: author's source

2. Backend

To connect the database to the frontend, we create a simple server, based on Node.js and Express.js. The server creates a connection to the MongoDB database through the Mongoose¹⁵ library, which represents the MongoDB object modeling tool. Mongoose requires that we define the schema of the collection in order to access it.

As shown in the Figure 34, we define the “Patients” variable that represents the collection of patients data. From this moment, each time we want to access this collection, we need to refer to the “Patients” variable.

¹⁵ Available at: <https://mongoosejs.com/> Accessed in July 2022

Figure 34: Mongoose schema definition

```
const patientsSchema = mongoose.Schema({ _id: Number, elements: {} });  
const Patients = mongoose.model("Patient", patientsSchema);
```

Source: author's source

We define two API services: get and update the patient graph. The detailed information is shown in Table 1.

Table 1: The API methods defined in the backend

HTTP method	Routing path	Mongoose function
GET	"/api/patient/:id"	Patients.findOne()
PUT	"/api/patient/save/:id"	Patients.findByIdAndUpdate()

Source: author's source

To get the patient graph, the user enters the corresponding patient ID. It is used to find the patient data into the database. The corresponding patient object is sent to the frontend, or an error message is returned in case the patient ID does not exist.

To update the patient graph, the patient ID and the updated graph are sent from the frontend. This new graph data entirely replaces the old one in the database.

3. Frontend

In order to access the described API, we use the Axios¹⁶ library. This library represents promise-based HTTP client.

To get the patient graph, the patient ID is sent to the API with the help of Axios library. In case of success, the graph is displayed. The user can now start working with the graph. In case of error, an alert window is shown, and the application is redirected to the home page (Figure 35).

¹⁶ Available at: <https://axios-http.com/> Accessed in July 2022

Figure 35: Fetch graph function

```
useEffect(() => {  
  async function fetchGraph() {  
    try {  
      let response = await axios.get(baseUrl + patientID);  
      setGraphElements(response.data.patient.elements);  
      setAllSubgraphs({  
        index: 0,  
        subgraphs: [response.data.patient.elements],  
      });  
    } catch (error) {  
      window.alert(error.response.data.msg);  
      navigate(`/homepage`);  
    }  
  }  
  fetchGraph();  
}, [patientID]);
```

Source: author's source

To update the graph, Axios send to the backend the patient ID as well as the graph to be saved (Figure 36). This function is called when the user clicks on the “Save changes” button. In both cases (success or error), the alert window displays a message, so the user knows if the changes were saved.

Figure 36: Update graph function

```
const saveGraphInDB = () => {  
  axios  
    .put(baseUrl + "save/" + patientID, graphElements)  
    .then((response) => {  
      window.alert("Changes saved!");  
    })  
    .catch((e) => window.alert("Server error! Cannot save the graph."));  
};
```

Source: author's source

3. DISCUSSION

In this chapter, the obtained results will be discussed. First, we will summarize the results of our study. Then, we will discuss the limits of the study and possible improvements. Finally, we will describe the recommendations for the future work.

3.1. Analysis of the results

The created application represents a simple and user-friendly tool for review and correction of the longitudinal tumor mapping graphs. The application allows the user to easily obtain information about each ROI and their connections. The visual features facilitate the overall perception of the graph and allow to track individual lesion trajectories. The reviewed connections between ROIs can be rapidly removed from the graph. The modified graph can be permanently saved at any time.

The use of Cytoscape.js library confirmed the correctness of the choice for the graph visualization. Cytoscape.js allowed to create a highly customized graph visualization and to easily control the graph entity. This library was integrated with the React in a simple way.

The JSON file format represents the basic format of the Cytoscape.js library as well as it is used in MongoDB. That is the reason, why the choice of that format allowed to avoid additional data cleaning and transforming processes.

The separation of the application architecture into three tiers will allow to easily change the backend technology and the database without having to perform major changes to the frontend. Such changes are necessary in case the application will be deployed in the hospital IT system.

3.2. Limitations

We have defined three main functionalities of our application: graph visualization, manipulation and storage. Although all three functionalities are presented in the application, we were mainly focused on the graph visualization part, since it had the prior value.

Currently, the graph saving process is not optimized, since in every update of the graph, the entire graph is saved and not only the changed elements.

Another limit concerns the graph manipulation process. For this moment, it is only possible to remove edges from the main graph. For future improvements, this functionality should be implemented also for the subgraphs.

Another limitation is related with the node colors, which represent the type of the ROIs. To distinguish suspicious and benign ROIs, we use red and green colors, respectively. However, such combination can be problematic for users with color blindness. For further development, it is recommended to provide a set of alternative colors or the custom color palettes.

The last point of improvement is related with the testing phase. Currently, the testing was limited by manual Alpha User Acceptance Testing. This type of testing allows to identify possible bugs and features before releasing the final product from the user point of view. The application was tested on the Chrome browser. For future improvements, different types of testing (functional, performance, regression etc.) should be performed on all types of browsers.

3.3. Future work

Although our application allows to visualize and review the tumor mapping graph, it still represents a prototype which demands future improvements. Our recommendations provide possible directions for future work:

- Image and segmentation data associated to each node. Currently, the user can observe the information about selected ROIs or its connections. For the future work, the user should have the possibility to observe the medical images of two connected ROIs in order to compare those images. Based on visual match, the user will approve or remove the automatically defined connection between those ROIs. The comparison of two medical images can be provided by the integration of an external tool into the application. One example of tool that allows the comparison of two images is MIM software¹⁷. The possibility to integrate such solution should be examined.

¹⁷ Available at: <https://www.mimsoftware.com/> Accessed in June 2022

- Reviewed elements. Currently, the visual representation of the graph does not reflect if a node or an edge were already reviewed. For further improvement, the “reviewed” property can be added to each element. This feature will help to visually style all reviewed elements differently than the others. Therefore, the user will easily distinguish the already analyzed ROIs from the ROIs that needs to be examined.
- Timeline. Currently, all ROIs of one timepoint are located in one column. That is how the user can visually distinguish the different timepoints. For future improvement, the timeline can be added in the background of the graph canvas. The points on the timeline will represent the date of imaging examinations.
- Graph manipulation. For now, the user can remove an edge to discard automatically assigned mapping. Further graph manipulation may include the creation of a new connection between the ROIs. Moreover, the possibility to add or change information of edges and nodes could be provided.
- Node positioning. Currently, the nodes position is based on the node data property “pos” where the X position is the date of imaging exam, and the Y position is the ROI identifier. This approach has limitations. If the difference of ROI identifiers is too high, the graph visualization is not comfortable to the user perception. For future improvement, other algorithms should be created. One possible solution is to define the nodes positioning based on the volume of the ROI, so the smaller ROIs will be located upper than the larger ROIs. Another solution is an algorithm that calculates the number of nodes of one timepoint and set equal distance between them.
- Threshold value. Currently, the threshold value removes the edges which have the “intersection mean” value is less than the chosen threshold value. For future development, the threshold can be used to remove not only the edges, but also nodes associated with these edges.
- Security. Future work assumes the creation of the user authentication due to security reasons. Currently, the application is deployed on the server of the university and provides limited access to the application.

- Session. The future improvements of the application should manage the user sessions. This will allow to store information of the user activities. If the user closes the browser and comes back, all the performed actions on the graph should be restored.
- Graph storage. When the user modifies and then saves the graph, the updated graph is directly saved in the database, without the possibility to retrieve the original graph. For the further work, this option may be provided. In addition, it would be useful to present the user the ability to continuously compare the original graph and the modified one. Another question that should be considered is the simultaneous work of two users on the same graph. If one user modifies and save the graph, how the modified graph will be sent to the second user? The response to this question will influence the architecture of the improved application.
- Testing. Currently, the graph features of the application were added based on the received suggestions and feedback from the research team. Future work should be aimed at testing the application by a "real user". Such testing will provide useful feedback about the usability of the application. Based on that feedback, the application can be significantly improved. The user may specify the functionalities that could be added to improve the graph review, such as node filtering, based on the volume, or selection of neighbors, based on the distance between ROIs.

The future improvements are not limited to this list of recommendations, but it can be a good starting point for those improvements.

4. CONCLUSION

This thesis aimed to create a web application that allows visualization, manipulation, and persistence of the automatically constructed graphs of individual patients. The obtained results indicate that the goal was achieved, but with some limitations.

The application represents a full stack solution with set of functionalities, that facilitate the review and correction of longitudinal tumor mappings. The developed tool allows the user to easily observe individual ROIs, to track the lesion trajectories, to correct the automatically created connections and to save the changes.

The created web application allows to confirm that the graph structure represents one of the optimal solutions for the visualization of longitudinal tumor mappings. Current existing applications for contouring tumor lesions allow to review automatically detected correspondences between pair of lesions. However, the review of each pair of lesions, instead of the entirety of connected lesions, is labor-intensive. Our application, based on the graph structure, demonstrates a significant optimization of the existing review process, as it considers the entirety of lesions in a time-based perspective.

However, future improvements of our prototype are needed in order to build a more robust application. Future work should take into account the feedback from other users, which can significantly influence the subsequent development process.

The results, obtained from our study, may be used for further development of the longitudinal mapping and review process. The enhanced application, based on our prototype, may be aimed to improve the existing automated mapping strategy, which is expected to have a positive impact on tracking the evolution of lesions.

REFERENCES

- Abler, D. (2022, 06 04). LongitudinalMapping_DataDescription_2022-05-04.docx. Retrieved 06 05, 2022
- Basler, L., Gabryś, H., Hogan, S., Pavic, M., Bogowicz, M., & Vuong, D. (2020, April 6). Radiomics, Tumor Volume, and Blood Biomarkers for Early Prediction of Pseudoprogression in Patients with Metastatic Melanoma Treated with Immune Checkpoint Inhibition. *CLINICAL CANCER RESEARCH*, pp. 4414-4426. doi:10.1158/1078-0432.CCR-20-0020
- Bi, W., Hosny, A., Schabath, M., Giger, M., & Birkbak, N. (2019, March/April). Artificial Intelligence in Cancer Imaging: Clinical Challenges and Applications. 69, pp. 127-157.
- Bojinov, V. (2016). *RESTful Web API Design with Node.js - Second Edition*. Packt Publishing.
- Bokeh. (2022). *Bokeh documentation*. Retrieved 07 05, 2022, from bokeh.org: <https://docs.bokeh.org/en/latest/>
- Brandes, U., Eiglsperger, M., & Lerner, J. (n.d.). *GraphML Primer*. Retrieved from GraphML Primer: <http://graphml.graphdrawing.org/primer/graphml-primer.html>
- Brath, R., & Jonker, D. (2015). *Graph Analysis and Visualization: Discovering Business Opportunity in Linked Data*. 544: Wiley.
- Butterfield, A., Ngondi, G., & Kerr, A. (2016). *A Dictionary of Computer Science (7 ed.)*. Oxford University Press. doi:10.1093/acref/9780199688975.001.0001
- Cai, J., Youbao, T., Yan, K., Harrison, A., Xiao, j., & Lin, G. (2021, April 12). Deep Lesion Tracker: Monitoring Lesions in 4D Longitudinal Imaging Studies. *arXiv:2012.04872v2 [cs.CV]*, pp. 1-17. doi:10.48550/arXiv.2012.04872
- Capobianco, N., Meignan, M., Cottureau, A.-S., Vercellino, L., Sibille, L., Spottiswoode, B., & Zuehlsdorff, S. (2021, January). Deep-Learning 18F-FDG Uptake Classification Enables Total Metabolic Tumor Volume Estimation in Diffuse Large B-Cell Lymphoma. *Journal of Nuclear medicine*, 62, pp. 30-36.

- Cytoscape.js. (2022). *Cytoscape.js*. Retrieved 07 10, 2022, from Cytoscape.js: <https://js.cytoscape.org/>
- Dash Cytoscape. (2019). *Dash Cytoscape*. Retrieved 07 05, 2022, from Plotly: <https://dash.plotly.com/cytoscape>
- Data-Driven Documents. (2022). *Data-Driven Documents*. Retrieved 07 10, 2022, from d3js.org: <https://d3js.org/>
- Erciyes, K. (2021). *Discrete Mathematics and Graph Theory. A Concise Study Companion and Guide*. Springer. doi:10.1007/978-3-030-61115-6
- Fass, L. (2008). Imaging and cancer: A review. *Molecular Oncology* 2, 115.
- Franz, M., Lopes, C., Huck, G., Dong, Y., Sumer, O., & Bader, G. (2016). Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics*, 32(2), pp. 309–311. doi:10.1093/bioinformatics/btv557
- Gehlenborg, N., & Wong, B. (2012). Networks. *Nature Methods*, 9, 115. doi:10.1038/nmeth.1862
- Gephi. (2022). *CSV Format*. Retrieved 07 05, 2022, from Gephi: <https://gephi.org/users/supported-graph-formats/csv-format/>
- Gephi. (2022). *GraphML Format*. Retrieved 07 05, 2022, from Gephi: <https://gephi.org/users/supported-graph-formats/graphml-format/>
- Gillies, R. J., Kinahan, P. E., & Hricak, H. (2016, February). Radiomics: Images Are More than Pictures, They Are Data. *Radiology: Volume 278: Number 2*, pp. 563-577.
- Gross, J., Yellen, J., & Anderson, M. (2019). *Graph Theory and Its Applications, Third Edition*. CRC Press.
- Gross, J., Yellen, J., & Zhang, P. (2013). *Handbook of Graph Theory, 2nd Edition*. Chapman and Hall/CRC.
- Ha, S., Choi, H., Paeng, J., & Cheon, G. (2019). Radiomics in Oncological PET/CT: a Methodological Overview. *Nuclear Medicine and Molecular Imaging*, pp. 14-29.

- Han, D., Pan, J., Zhao, X., & Chen, W. (2021). NetV.js: A web-based library for high-efficiency visualization of large-scale graphs and networks. *Visual Informatics* 5, pp. 61-66. doi:10.1016/j.visinf.2021.01.002
- Kapoor, A. (2021, 11 18). *Benefits of Using MERN Stack*. Retrieved 07 19, 2022, from Enlear Academy: <https://enlear.academy/benefits-of-using-mern-stack-7e0c732b5214>
- Koroliova, E. W. (2018). *MERN Quick Start Guide*. Packt Publishing.
- Meng, Y., Sun, J., Qu, N., Zhang, G., Yu, T., & Piao, H. (2019, 11). Application of Radiomics for Personalized Treatment of Cancer Patients. *Cancer Management and Research*, pp. 10851–10858.
- MongoDB. (2022). *MERN Stack Explained*. Retrieved 07 19, 2022, from MongoDB: <https://www.mongodb.com/mern-stack>
- Moreau, N., Rousseau, C., Fourcade, C., Santini, G., Brennan, A., Ferrer, L., & Lacombe, M. (2022). Automatic Segmentation of Metastatic Breast Cancer Lesions on 18F-FDG PET/CT Longitudinal Acquisitions for Treatment Response Assessment. *Cancers* 14(1), 101, pp. 1-16.
- NetworkX. (2022). *NetworkX*. Retrieved 07 05, 2022, from <https://networkx.org/>
- Pyvis. (2018). *Introduction*. Retrieved 07 05, 2022, from Pyvis: <https://pyvis.readthedocs.io/en/latest/introduction.html>
- Rahman, M. (2017). *Basic Graph Theory*. Springer.
- Sigma.js. (2022). *sigma.js*. Retrieved 07 10, 2022, from <https://www.sigmapjs.org/>
- Thakkar, M. (2020). *Building React Apps with Server-Side Rendering: Use React, Redux, and Next to Build Full Server-Side Rendering Applications*. Apress. doi:10.1007/978-1-4842-5869-9_3
- Vis.js. (2022). *Vis.js*. Retrieved 07 10, 2022, from Visjs.org: <https://visjs.org/>
- Wang, R., Perez-Riverol, Y., Hermjakob, H., & Vizcaino, J. (2015). Open source libraries and frameworks for biological data visualisation: A guide for developers. *Proteomics*, 15, pp. 1356–1374. doi:10.1002/pmic.201400377

APPENDIX I: COMPARATIVE ANALYSIS OF GRAPH VISUALIZATION LIBRARIES

Library	Language	Open-source	Web-based	Customization	Interaction	Documentation	Huge number of nodes	Stars in GitHub	Website
NetworkX	Python	+++	+++	++	-	+++	+	11 000	https://networkx.org/
Pyvis	Python	+++	+++	+++	++	++	+++	518	https://pyvis.readthedocs.io/en/latest/
Bokeh	Python	+++	+++	+++	++	++	+++	16 500	https://bokeh.org/
Dash cytoscape	Python	+++	+++	+++	+++	++	+++	446	https://dash.plotly.com/cytoscape
Vis.js	JavaScript	+++	+++	++	++	+++	+++	2 100	https://visjs.org/
D3.js	JavaScript	+++	+++	+++	++	+	+	102 000	https://d3js.org/
Sigma.js	JavaScript	+++	+++	+	++	-	+++	10 000	https://www.sigmapjs.org/
Cytoscape.js	JavaScript	+++	+++	+++	+++	+++	+++	8 500	https://js.cytoscape.org/

Source: author's source


Excellent: +++, Good: ++, Basic: +, Missing : - ;

APPENDIX II: ALL COMPLETED USER STORIES IN THE PRODUCT BACKLOG

US Nr	Theme	As an/a ...	I want to ...	so that ...	Acceptance Criteria	Blockers / Dependencies	Priority	Status	SP	Sprint	US accepted	MoSCoW
1	Research	Developer	Do the research on the topic	I understand better the context	Understand the topic and the context		1000		5	0	17.05.2022	Must
2	Preparation	Developer	Work on the mockups	I can better visualise the futur application	The mockup is created		950		8	0	17.05.2022	Must
3	Preparation	Developer	Explore the set of provided data	I can better choose the technologie	Understand the data		900		1	0	17.05.2022	Must
4	Research	Developer	Do the research on existing graph visualisation libraries	I can use it to visualise the graphs	Few libraries tested, the most suitable ones are chosen		800		21	1	31.05.2022	Must
5	Research	Developer	Choose the relevant data format for the graph	I can process the graph data easily	Raw data are converted in the right format, so it can be used then to diplay in the graph		750		8	2	14.06.2022	Must
6	Graph	User	Visualize the graph of one patient	I can see all POI of one patient	The graph is shown as in mockup	Use of local data	700		13	2	14.06.2022	Must
7	Graph	User	Zoom the graph	I can see the details	The graph can be zoomed/dezoomed		650		5	2	14.06.2022	Must
8	Graph	User	Have visual features of the graph (labels, colors, etc)	I can easily interact with the graph	The graph has features: node and edge labels, colors, edge thickness		625		13	3	28.06.2022	Must
9	Graph	User	Interact with graph	I can see additional information about selected element (node or edge)	When a node or an edge selected, addition information is shown		600		5	3	28.06.2022	Must
10	Graph	User	Set the threshold on the edge values	I can filter the lesions trajectories	The slider component which filters the edge values		575		8	3	28.06.2022	Must
11	Graph	User	Select all related POI	I can explore a subgraph of individual lesions trajectories	The subgraph is selected and shown separately of the general graph	US 10	550		13	4	12.07.2022	Must
12	Graph	User	Remove the edge between two nodes	I can discard the automatically assigned mapping	Selected edge is removed from those nodes		525		8	4	12.07.2022	Must
13	GUI	User	I want to see the graph of certain patient	I can analyse the lesions of that patient	A window with input to enter the information of particular patient and receive back the individual graph. The response is sent by API		500		21	5	21.07.2022	Must
14	GUI	User	I want to save the changes that I've made during my connection	I can save the changes definitively.	The changes in node connections are saved via API		475		13	5	21.07.2022	Must

Source: author's source

APPENDIX III: EXAMPLE OF SPRINT BACKLOG

				Sprint	3						
		Story points									
8	Have visual features of the graph (labels, colors, etc)	13	Start date	14.06.2022							
9	Interact with graph	5	End date (included)	27.06.2022							
10	Set the threshold on the edge values	8	Initial sprint estimation	26							
			Sprint goal	Have visual features and filters on the graph							
			Nbr working days	6							
ID US.task	Task Name	Resp.	Initial Estimate		Day 1 14.06.2022	Day 2 15.06.2022	Day 3 20.06.2022	Day 4 21.06.2022	Day 5 22.06.2022	Day 6 27.06.2022	
8.1	Create a basic graphical structure of web application: sidebar, panels, buttons, etc.	Ekaterina	14		14	7	0	0	0	0	
8.2	Add edge labels, edge thickness, node colors	Ekaterina	5		5	5	3	0	0	0	
8.3	Add node volume visualisation	Ekaterina	6		6	6	6	2	0	0	
9.1	Display the information of element (node of edge) when it is selected	Ekaterina	4		4	4	4	4	0	0	
9.2	Filter nodes based on it's type	Ekaterina	5		5	5	5	4	2	0	
10.1	Add a threshold slider to manage threshold settings of the graph	Ekaterina	8		8	8	8	8	6	0	
		Total	42								
		Remaining	42		42	35	26	18	8	0	
		Theoretical	42		42,0	33,6	25,2	16,8	8,4	0,0	

Source: author's source

AUTHOR'S DECLARATION

I hereby declare that I have carried out this final research project on my own without any help other than the references listed in the list of references and that I have only used the sources mentioned. I will not provide a copy of this paper to a third party without the permission of the department head and of my advisor, including the partner company with which I collaborated on this project, with the exception of those who provided me with information needed to write this paper and whose names follow:

- Mr. Adrien Depeursinge,
- Mr. Roger Schaer,
- Mr. Daniel Abler.

Sierre, the 27th of July 2022

Ekaterina Aymon