FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of master's thesis

Title:	The Development of a New Visualization Tool for the IKEM
Student:	Bc. Karel Vrabec
Supervisor:	Ing. Petr Pauš, Ph.D.
Study program:	Informatics
Branch / specialization:	Software Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2024/2025

Instructions

The IKEM is one of the most prominent medical facilities for healthcare, treatment, and scientific research in the Czech Republic, known worldwide. It is specialized in cardiology, diabetes, and human organ transplantation.

Doctors use a magnetic resonance imaging (MRI) machine to see the human organs and body from the inside. Scientists from the IKEM search for new methods and improvements in this field. They use ParaView as a special software tool for working with MRI machines, but it is difficult, complicated, and overwhelming for both groups.

The goal is to develop a brand new web application as a layer lying above ParaView, reducing its redundant functionalities and simplifying the whole work process. In cooperation with the doctors and scientists from the IKEM, study the domain and analyze all software requirements. Design and implement the web application accessible in the web browser with the help of modern web technologies. Focus on the user interface and test its functionalities and usability with relevant users.

This assignment arose as a result of mutual cooperation between the experts from the IKEM in Prague, CZE, and the University of Texas Southwestern Medical Center in Dallas, USA. Its successful realization might be very helpful to scientists, doctors, and eventually patients.

Electronically approved by Ing. Michal Valenta, Ph.D. on 8 February 2024 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

The Development of a New Visualization Tool for the IKEM

Bc. Karel Vrabec

Department of Software Engineering
Supervisor: Ing. Petr Pauš, Ph.D.

May 8, 2024

Acknowledgements

I would like to thank my supervisor, Ing. Petr Pauš, Ph.D., for the leadership, time, and advice he gave me while I was working on this master's thesis. Another huge thanks goes to Ing. Kateřina Škardová, Ph.D., and doc. Ing. Tomáš Oberhuber, Ph.D., for their great help, clarifications, and explanations of the client's requirements and needs. Next, I am really grateful to Dr. Sandeep Kuttal for being my advisor while I was studying at North Carolina State University in the USA. Last but not least, I would like to thank Ing. Jakub Klinkovský, Ph.D., for the deployment of the newly created web application and making it accessible for the users. Thank you very much, Sébastien Jourdain from Kitware, for your technical and professional help while I was struggling with modern web technologies. Thanks to all the testers for participating in the usability testing, my family for their support, and other great people for encouraging me.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on May 8, 2024 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Vrabec, Karel. *The Development of a New Visualization Tool for the IKEM.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024.

Abstrakt

Tato diplomová práce se zabývá tvorbou nové webové aplikace pro pohodlnější práci s výstupy z magnetické rezonance (MRI). Zadavatelem je Institut klinické a experimentální medicíny (IKEM), české zdravotnické zařízení známé díky svému zaměření na transplantaci lidských orgánů, diabetes a kardiovaskulární choroby, jejich léčbu a vědecký výzkum. Výzkumníci z IKEM používají ParaView, populární nástroj pro komplexní a interaktivní vizualizace, pro prohlížení MRI výstupů. Prostředí tohoto programu je pro ně však velice komplikované. Kromě toho je nástroj ParaView přístupný pouze jako desktopová aplikace. Pomocí moderních technologií, jako jsou Trame, VTK a Vuetify, je navrženo a vyvinuto zcela nové řešení. Výsledkem práce je kontejnerizovaná, nasaditelná webová aplikace, která poskytuje nové interaktivní a zjednodušené uživatelské rozhraní pro snadnou správu, prohlížení a interakci s MRI výstupy dle požadavků zadavatele.

Klíčová slova webová aplikace, IKEM, magnetická rezonance, vizualizace, ParaView, UI, Python, Trame, VTK, Vuetify

Abstract

This master's thesis deals with the development of a brand new web application for better and more convenient work with outputs from magnetic resonance imaging (MRI) machines. The web application was created for the Institute for Clinical and Experimental Medicine (a client), a Czech medical facility well known for its focus on human organ transplantation, diabetes and cardiovascular therapies, treatments, and scientific research. The researchers of the IKEM (a user) use ParaView, a popular tool for complex and interactive visualizations, for viewing MRI outputs. However, the environment of ParaView is very complicated. Additionally, the user is able to access ParaView as a desktop application only. Therefore, in this thesis, all user's work processes and requirements are analyzed. Finally, a new solution is designed and developed with the help of modern technologies, such as Trame, VTK, and Vuetify. The result of this work is a containerized, deployable web application that provides a new interactive and simplified user interface to manage, view, and control MRI outputs easily.

Keywords web application, IKEM, magnetic resonance imaging, visualization, ParaView, UI, Python, Trame, VTK, Vuetify

Contents

Introduction **1**

1 Goal **3**

2 Context **5**
2.1 IKEM 5
2.2 Magnetic Resonance Imaging 6
2.2.1 Principle 7
2.2.2 4D Flow MRI 8
2.3 ParaView 8
2.4 Medical File Formats 9
2.4.1 DICOM 9
2.4.2 VTI 10

3 Analysis **11**
3.1 Previous State 11
3.1.1 Workflow in ParaView 12
3.2 Desired State 14
3.2.1 Functional Requirements 15
3.2.2 Non-Functional Requirements 16
3.2.3 Use Cases 17
3.2.4 Use Case Diagram 22
3.3 Similar Solutions 23
3.3.1 Visualizer 23
3.3.2 LightViz 24
3.3.3 ArcticViewer 24
3.3.4 Decision 24
3.4 Technical Ways 24
3.4.1 WebGL 24
3.4.2 Three.js 25

3.4.3 ParaViewWeb . . . 25
3.4.4 Trame . . . 25
3.4.5 Decision . . . 26

4 Design **27**
4.1 MRI Viewer . . . 27
4.2 Personas . . . 27
4.2.1 Persona A . . . 28
4.2.2 Persona B . . . 28
4.2.3 Persona C . . . 29
4.3 Wireframes . . . 29
4.4 Prototype . . . 31

5 Implementation **35**
5.1 Technologies . . . 35
5.1.1 Trame . . . 35
5.1.1.1 MVVM Pattern . . . 36
5.1.1.2 Cookiecutter . . . 37
5.1.2 VTK . . . 37
5.1.2.1 VTK.js . . . 38
5.1.2.2 Visualization Example . . . 38
5.1.3 Vuetify . . . 40
5.1.3.1 Usage in Trame . . . 40
5.2 Development . . . 41
5.3 Web Application . . . 42
5.3.1 Engine . . . 42
5.3.2 File Manager . . . 43
5.3.3 Language Manager . . . 43
5.3.4 VTK Manager . . . 43
5.3.4.1 Interaction . . . 44
5.3.4.2 Picking . . . 44
5.3.4.3 Rendering . . . 44
5.3.4.4 Slicing . . . 44
5.3.5 Architecture . . . 45
5.3.6 User Interface . . . 46
5.3.6.1 Components . . . 46
5.3.7 Design Patterns . . . 48
5.3.7.1 Decorator . . . 49
5.3.7.2 Facade . . . 50
5.3.7.3 Singleton . . . 51

6 Testing **53**
6.1 Tests . . . 53
6.2 Test Data . . . 53

6.3 Compatibility 54
6.4 Usability 55
6.4.1 Results 55

Conclusion **57**

Bibliography **59**

A Acronyms **63**

B User Guide **65**
B.1 MRI Viewer 65
B.2 Features 65
B.2.1 Upload Files 66
B.2.1.1 Limitations 66
B.2.1.2 Error Codes 66
B.2.1.3 Recommendations 67
B.2.2 Manage Files 68
B.2.3 File 68
B.2.3.1 Groups 69
B.2.3.2 Memorization 69
B.2.4 Data Array 69
B.2.5 Representation 69
B.2.6 Color Map 70
B.2.7 Interaction 70
B.2.7.1 Slice 71
B.2.7.2 Zoom 72
B.2.7.3 Translation 72
B.2.7.4 Rotation 73
B.2.8 Player 73
B.2.8.1 Recommendations 74
B.2.9 Point and Cell Information 74
B.2.10 Axes Information 75
B.2.11 Axes Widget 76
B.2.12 Scalar Bar 77
B.2.13 Reset View 77
B.2.14 Dark and Light Themes 77
B.2.15 Languages 78
B.2.16 Progress Bar 78
B.3 Contact 79

C Test Scenarios **81**
C.1 Language 81
C.2 Slice Tool 83

C.3 Player 84
C.4 Picking 86

D Contents of CD **89**

List of Figures

2.1 The logo of the IKEM 5
2.2 The MRI machine 6
2.3 Visualization in ParaView 9

3.1 Activity diagram showing the workflow in ParaView 5.12 12
3.2 The overview of functional requirements 15
3.3 The overview of non-functional requirements 16
3.4 Mapping requirements to use cases 17
3.5 The overview of use cases 18
3.6 Use case diagram 22
3.7 Visualization in Visualizer 23

4.1 Loading options 29
4.2 Working environment 30
4.3 Cell information 31
4.4 Dialog for uploading files 32
4.5 Working environment 32
4.6 Selects 33
4.7 Upper bar 33
4.8 Tools 33

5.1 The MVVM pattern of Trame 36
5.2 The output of the visualization example 40
5.3 The human brain visualized in *MRI Viewer* 42
5.4 The architecture of the web application 45
5.5 The architecture of the user interface 46

6.1 The blood flow in the aorta visualized in *MRI Viewer* 54

B.1 A dialog for uploading files 66
B.2 An error while uploading files 67

B.3 A dialog for managing files 68
B.4 The list of files 68
B.5 The list of data arrays 69
B.6 The list of representations 70
B.7 The list of color maps 70
B.8 Slice tool 71
B.9 A slice example 71
B.10 Zoom tool 72
B.11 Translation tool 72
B.12 Rotation tool 73
B.13 Player 73
B.14 Point and cell information 74
B.15 Point information 74
B.16 Cell information 75
B.17 Axes information 75
B.18 An axes information example 76
B.19 Axes widget 76
B.20 Scalar bar 77
B.21 Reset view 77
B.22 Theme 78
B.23 Languages 78
B.24 Progress bar 79

C.1 Starting point in the first test scenario 82
C.2 Ending point in the second test scenario 84
C.3 Ending point in the third test scenario 86
C.4 Picking a random point in the data 88

List of Codes

2.1 The basic XML structure of the VTI file 10
5.1 The basic visualization in VTK 39
5.2 The Decorator design pattern in *MRI Viewer* 49
5.3 The Facade design pattern in *MRI Viewer* 50
5.4 The Singleton design pattern in *MRI Viewer* 51

Introduction

Magnetic resonance imaging (abbreviated as "MRI") is commonly known as a clinical and medical method for examination and visualization of the internal human body. MRI has evolved at a high pace since its very beginning in the last century. In today's world, it is a commonly used technique accessible in the form of robust MRI machines in any major hospital.

IKEM is a healthcare facility that possesses and uses MRI machines for medical and research purposes. The researchers at the IKEM are able to view outputs from MRIs through the ParaView visualization tool. With the help of the four-dimensional flow MRI technique, they can examine blood flow in the vessels and diagnose possible anomalies. The technique encodes 3D images of velocities in time. Although it was already invented in the last century, the 4D flow MRI method was practically used only recently due to a lack of hardware performance.

However, there is an omnipresent problem with the availability and usability of these modern scientific achievements. ParaView has an overwhelming and complex environment full of many various features. Therefore, the process of loading, viewing, tuning parameters, and examining MRI outputs is difficult in this environment. If the researchers have a problem with complexity, doctors without any knowledge of ParaView and similar visualization tools will have the problem, too. That is the main reason for creating a new lightweight solution that will be conveniently available in the web browser and usable by researchers and, in the future, doctors. It may help support not only medical research but also healthcare in the Czech Republic.

In the theoretical part of this thesis, the reader is introduced to the IKEM, the principles and specific types of MRI (i.e., 4D flow MRI), and the ParaView visualization tool. This part of the work also contains a brief description of DICOM and VTI as the file formats commonly used in healthcare. In the second half of the part, an analysis of the previous and desired states is conducted. The client's requirements, use cases, user's work process, similar solutions, and technical ways to solve the problem are described.

In the practical part of the thesis, the reader can continue with three chapters. The first chapter focuses on the web application's design, as described with the help of wireframes and a final prototype. The second chapter describes the implementation of the web application with the help of modern web technologies, such as Trame. This part of the work also contains the definitions of all necessary terms and a description of these technologies. Finally, the last chapter presents testing in terms of functionalities, compatibility, and usability. All chapters are supplemented with auxiliary images and diagrams.

Chapter 1

Goal

The main goal of this master's thesis is to develop a containerized, deployable, and lightweight web application according to the client's requirements and needs. The web application features a new interactive and simplified user interface for managing, visualizing, and controlling MRI outputs. The selected distribution type (i.e., a web application), less complexity (i.e., only desired and important functionalities), and appropriate modern web technologies together achieve this goal. Therefore, to accomplish it systematically, the thesis has several smaller aims.

The first aim is to analyze the client's requirements and needs with the help of researchers from the IKEM. This also includes getting to know the client and the respective domain.

The second aim is to design and implement the web application with the help of modern web technologies (with a focus on the user interface). It is important to develop a usable and clear UI that is understandable even for non-technical users.

The last aim is to test the web application's features, compatibility, and usability (with relevant users). Additionally, there is an optional aim to deploy the web application using Docker with the help of engineers from the IKEM.

Chapter 2

Context

This chapter provides background for a reader to become familiar with the topic. It contains brief descriptions of the IKEM (as the client), MRI and its principle, ParaView (as the visualization tool), and two medical file formats relevant for the following chapters.

2.1 IKEM

IKEM is one of the specialized medical care facilities in the Czech Republic. The abbreviation stands for the Institute for Clinical and Experimental Medicine. The institution was established in the early 1970s and is located in Prague. The local staff is composed of doctors, nurses, paramedics, scientists, researchers, and collaborators from other institutions, like universities. [1]

Its goal is to enhance and evolve health care, therapies, and treatments to make the whole process much better, more modern, and more pleasant for patients. The main focus is on solving the most serious diseases and performing ambitious acts in healthcare. These are primarily cardiovascular problems (i.e., heart and vascular diseases), diabetes (i.e., metabolic disorders), and the transplantation of human organs. In addition, the IKEM is also a scientific research center, which makes it possible to put the latest knowledge and discoveries into practice. [1]

Figure 2.1: The logo of the IKEM [2]

2.2 Magnetic Resonance Imaging

Magnetic resonance imaging (abbreviated as “MR” or “MRI”) is a modern, non-invasive medical technology, a method, and a tool for displaying human organs and tissues. It is used to reveal, diagnose, and localize diseases, anomalies, and defects such as tumors. MRI was put into practice in the 1970s based on the discoveries by Nobelists Paul Lauterbur and Peter Mansfield [3]. [4] [5]

The method is safe, precise, painless, and reliable. Unlike a CT (that is, “Computed Tomography”) scan, there is no ionizing radiation or x-rays. It is used in different branches of medicine, such as neurology, cardiology, or oncology. With the help of MRI, doctors can examine, for example, the brain, spinal cord, spine, heart, liver, kidneys, pancreas, stomach, knees, shoulders, and so forth. [4] [5]

An MRI machine is a huge magnet in the shape of a tunnel. It also includes a movable bed, a service stand, and a processing system. The machine is operated by a specialized radiology technician. During the examination, a patient is lying still on the bed that is moved inside the MRI machine. The work of the MRI machine is accompanied by a loud noise. Therefore, headphones are usually available. There is also a communication device so that the patient remains in contact with the technician. Nowadays, the whole process takes 15 minutes, sometimes up to an hour or even more (depending on the specific procedure). Thus, the examinations and the work of MRI machines can be time-consuming. [4] [5]

Figure 2.2: The MRI machine [6]

The MRI machine scans and slices a specific part of a human body. These slices can then be merged together to create a detailed 3D image. To improve the quality of the output, contrast substances (such as gadolinium) can be applied. A DICOM (that is, "Digital Imaging and Communications in Medicine") format is used for storing the results of the MRI examination [7]. These results are evaluated and interpreted by a radiologist, who sends them to a treating doctor within several days. [4] [5]

The purchase, installation, and service of the MRI machine are very expensive, including related operational costs (such as power consumption or maintenance). Therefore, only major hospitals, centers, or facilities usually possess these machines. [8]

The patient should not have any type of metal gear in or on his or her body while being examined with the MRI machine. It can be dangerous to conduct the examination if the patient's body contains metal implants or devices, such as cardiac pacemakers. Last but not least, such items can distort and misrepresent final images. Additionally, pregnancy and claustrophobia may be problematic, too. Either way, the situation should be assessed by a doctor first. There is no problem with braces or dental fillings. [4] [5]

2.2.1 Principle

MRI is based on a very strong static magnetic field (with the power of several Tesla units) and electromagnetic waves. When the patient is being examined with the help of the MRI machine, it sends radio-frequency impulses to disrupt the strong magnetic field. Afterwards, it receives and decodes signals from the particles in the patient's body that react to these external changes in the outer magnetic field. [5] [9] [10]

The core of an atom contains protons and neutrons. Protons rotate around their axes, which is generally called a spin. Together with their natural positive electrical charges, the core can generate a small magnetic field. The human body is composed of water, which is present in every human cell. The water contains hydrogen, which has a strong magnetic potential. [9] [10] [11]

The MRI machine creates a strong static magnetic field around a patient's body. This field influences and aligns the protons' spins to rotate around their axes in the direction of the outer magnetic field (or opposite). Then, radio-frequency pulses are applied perpendicularly to the magnetic field to briefly disturb the protons out of alignment. These disruptions create signals that are captured by special sensors. Received signals are different according to the type of tissue. In the end, they are converted to digital images. [5] [9] [10]

2.2.2 4D Flow MRI

MRI has wide applications in healthcare, where it can be used to diagnose, for example, cardiovascular diseases. Such an MRI application is called CMRI (that is, "Cardiac Magnetic Resonance Imaging") [12].

Phase-contrast MRI (abbreviated as "PC-MRI") is a variant of MRI used to obtain velocities of the flow, such as the blood flow in the case of CMRI. Velocity is the speed and direction of a moving object. PC-MRI can be further divided into 2D and 4D flow MRI. [13] [14]

2D flow MRI is an older type of PC-MRI, where the velocity is encoded in two dimensions (i.e., one direction and time) at every node of a plane. On the other hand, 4D flow MRI is a more recent extension of PC-MRI, enabling a more comprehensive understanding of the blood flow. The velocity is encoded into three dimensions (x, y, and z) at every node of a space throughout time (i.e., a sequence of MRI datasets). Then, one MRI dataset contains three velocity components. [13] [14]

4D flow MRI is typically used in healthcare for imaging and examining the blood flow in vessels, veins, and arteries, especially in the human heart and aorta. However, due to a lack of performance, high complexity, and a long processing period, it is still not usual for clinical use. [13] [14]

2.3 ParaView

ParaView is complex, cross-platform, and open-source software for the analysis, exploration, and visualization of large datasets. It started in 2000 as a shared project between Kitware (as the author and developer of ParaView) and Los Alamos National Laboratory. The tool is able to display datasets and specific simulations in many scientific domains, such as aerospace, astrophysics, climate science, fluid dynamics, medical science, mechanical engineering, structural analysis, and so forth. Nowadays, ParaView is a common tool in many laboratories, universities, and the commercial sector. [15]

Besides visualizing the data, it has plenty of customizable capabilities and techniques. These are especially coloring, slicing, clipping, plotting, volume rendering, animation, data selection and highlighting, and many other features for gaining insight into the data. Furthermore, ParaView can be extended by creating Python scripts. It also has extensive support for different file formats, such as DICOM or VTI (for visualizing CT and MRI data). [15]

ParaView can be downloaded as a desktop application, along with the documentation and test data. On top of that, it can be run in client/server mode to perform remote parallel computation on HPC (that is, "High-Performance Computing"). The client serves as a data viewer, while the server runs as an MPI (that is, "Message Passing Interface") process that works with the data. Kitware supports the ParaView community extensively by creating tutorials, webinars, and training courses. [15]

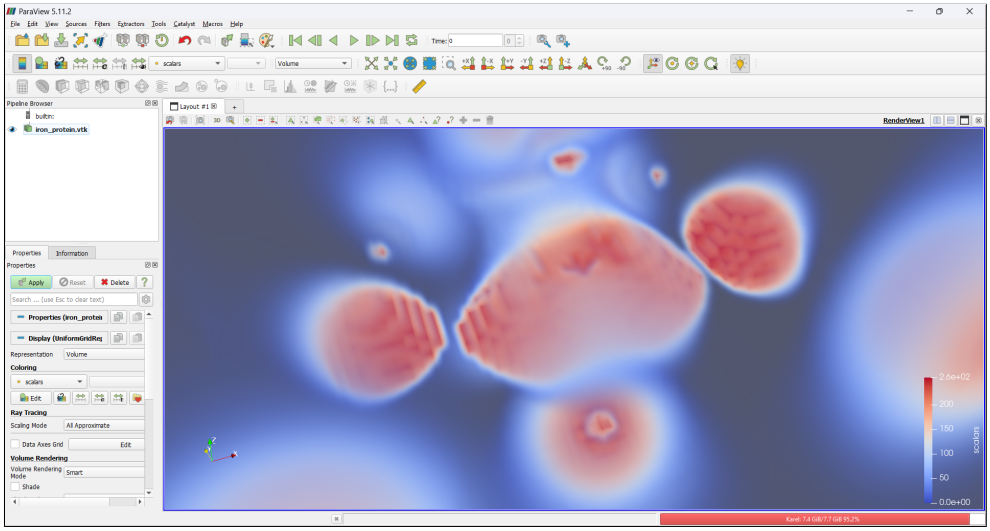

Figure 2.3: Visualization in ParaView

2.4 Medical File Formats

Although there are more medical file formats, such as Analyze, NIfTI, or MINC, only DICOM and VTI are briefly described below, as they will be useful for the coming chapters. In fact, VTI is not considered a pure medical file format, but it is also used in the field of medical science. [16]

2.4.1 DICOM

Digital Imaging and Communications in Medicine (abbreviated as "DICOM") is a standard file format for storing medical images used in healthcare and medical research. It was initially published in 1993. Nowadays, the usage of this file format is very widespread. DICOM is widely used in many healthcare domains, such as radiology (i.e., X-ray, CT, ultrasound, and MRI devices), cardiology, veterinary medicine, dentistry, and so forth. For instance, DICOM is used for storing raw outputs from the MRI machines. [16] [17]

A DICOM file contains metadata and image data. The header of the file includes the patient's information, such as name, gender, age, and so on. Generally, DICOM allows doctors to make faster diagnoses. Therefore, patients can be treated effectively and quickly. It also helps in terms of data compatibility, exchange, and unification. [16] [17]

2.4.2 VTI

Visualization Toolkit Image Data (abbreviated as "VTI") is a file format typically associated with ParaView, which is a visualization tool built upon VTK (that is, "Visualization Toolkit"). It uses the XML (that is, "eXtensible Markup Language") syntax to store the image data and related information. Additionally, the VTI format has support for compression, various encodings, and byte order. [18] [19]

Essentially, VTI is a serial and structured file format. The operations of reading and writing are executed by a single process, and the data is included in only one file. Moreover, the dataset is a regular grid of the image data. By default, there are two kinds of data (i.e., points and cells) to be defined in each of the VTI files. [18] [19]

```
<VTKFile type="ImageData" ...>
    <ImageData WholeExtent="x1 x2 y1 y2 z1 z2"
     Origin="x0 y0 z0" Spacing="dx dy dz">
        <Piece Extent="x1 x2 y1 y2 z1 z2">
            <PointData>...</PointData>
            <CellData>...</CellData>
        </Piece>
        ...
    </ImageData>
</VTKFile>
```

Code 2.1: The basic XML structure of the VTI file [19]

Chapter 3

Analysis

In this chapter, the reader is introduced to the previous state, when the researchers worked with ParaView. It includes all the problems the researchers had and a description of a workflow with typical actions they used to do while working with ParaView. Afterwards, the desired state, including all the requirements and use cases, is described. Finally, an analysis of similar solutions and technical ways to solve the researchers' problems is conducted. Then, the next chapter focuses on the proposed solution.

3.1 Previous State

While working with an MRI machine and receiving 4D flow MRI data, researchers converted these data from the DICOM format to VTI. The reason was that the DICOM format and its file viewers were not appropriate and user-friendly for them. Besides, the data were also anonymized and denoised. Finally, researchers used ParaView to visualize the exported *.vti* file. However, ParaView was not accepted by the doctors because of its complexity.

Researchers at the IKEM use ParaView as a desktop application. It is necessary to install the software directly on the computing machine. Of course, this can be very limiting, binding users to specific computers. Unfortunately, ParaView is also a huge software with a complex user interface containing all possible features. Beginning users may have problems familiarizing themselves, getting oriented, and working in this complex environment, even if they are technically talented. The main reason is the large number of functionalities, icons, tools, shortcuts, and so forth. Typically, researchers need to use only basic tools, such as slicing.

Moreover, working with the input data in ParaView at the very beginning is difficult, time-consuming, and consists of many clicks. In addition, the process of loading and researching the data is very regular and repetitive for the researchers. Kitware (as the author of ParaView) does offer support services to customize ParaView, but they are pre-paid and expensive [20].

This status quo is neither appropriate for the researchers nor the doctors. Therefore, it is necessary to create an effective, free, and lightweight solution containing only the capabilities and features that are really necessary.

3.1.1 Workflow in ParaView

Our user is a researcher from the IKEM who works with ParaView. While working with the program, the user wants to achieve his or her goals (e.g., to see the blood flow in the aorta over time). The user's goals can be accomplished by going through a work process that is composed of different actions. By revealing this work process, the user's goals and needs can be better understood. Therefore, the new solution contains only the features that are really necessary, whereas other features are filtered out. The work process of the researcher in ParaView is depicted in the following activity diagram, which was discussed and approved by the client. All activities are described below.

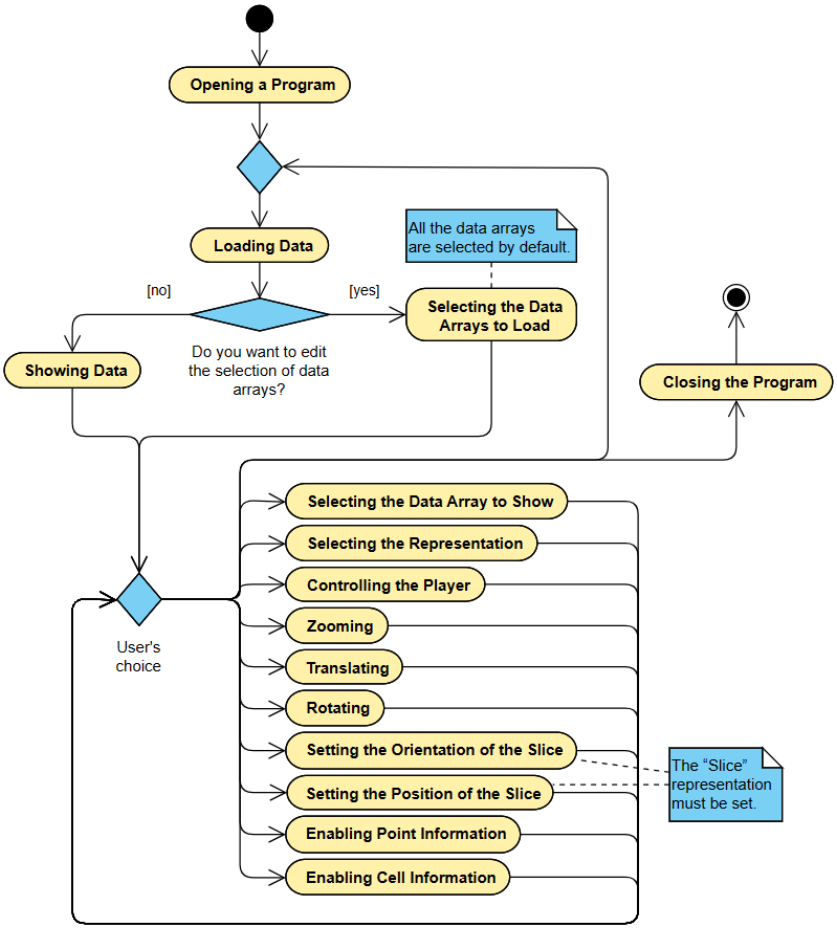

Figure 3.1: Activity diagram showing the workflow in ParaView 5.12

Shortcut: A1
Title: Opening a Program
Description: The user opens ParaView.

Shortcut: A2
Title: Loading Data
Description: The user clicks *File* → *Open* and selects the data (i.e., VTI files). Finally, he or she approves by clicking *OK*.

Shortcut: A3
Title: Showing Data
Description: The user clicks on the *eye* icon to show the data. The default representation is *Outline*.

Shortcut: A4
Title: Selecting the Data Arrays to Load
Description: VTI files are composed of different point or cell data arrays. The user checks or unchecks the checkboxes depending on what data arrays he or she wants to load. Finally, he or she approves by clicking *Apply*, which also shows the data. The default representation is *Outline*.

Shortcut: A5
Title: Selecting the Data Array to Show
Description: The user opens the respective menu and selects the data array to visualize.

Shortcut: A6
Title: Selecting the Representation
Description: The user opens the respective menu and selects the representation (e.g., *Points*, *Slice*, *Surface*, *Volume*, etc.).

Shortcut: A7
Title: Controlling the Player
Description: The user clicks to play, pause, or skip to the previous or next file. The group of VTI files is played only once by default, unless the user clicks the *loop* icon.

Shortcut: A8
Title: Zooming
Description: The user zooms the data in or out with the help of the mouse.

Shortcut: A9
Title: Translating
Description: The user translates the data with the help of the mouse.

Shortcut: A10
Title: Rotating
Description: The user rotates the data with the help of the mouse.

Shortcut: A11
Title: Setting the Orientation of the Slice
Description: The user opens the respective menu in the left column and selects the slice orientation (i.e., *XY*, *YZ*, or *XZ*).

Shortcut: A12
Title: Setting the Position of the Slice
Description: The user moves with a slider or types a value in the left column to change the slice position.

Shortcut: A13
Title: Enabling Point Information
Description: The user clicks on the *hover points on* icon to show the point information (i.e., an identificator, coordinates, and the values of point data arrays at that particular point) while hovering over points.

Shortcut: A14
Title: Enabling Cell Information
Description: The user clicks on the *hover cells on* icon to show the cell information (i.e., an identificator and the values of cell data arrays at that particular cell) while hovering over cells.

Shortcut: A15
Title: Closing the Program
Description: The user closes ParaView.

3.2 Desired State

An ideal solution is to create a new web application with only the relevant features (such as the visualization of VTI files). It solves the complexity and availability problems. Unnecessary functionalities are removed, including redundant steps (clicks) in the workflow above (such as *A3*, *A4*, or setting *Outline* as the default representation). The web application is also accessible through the web browser. Additionally, the user support is enhanced with multilingual user documentation, tooltips, or information icons.

The web application is intended for use by researchers first rather than doctors in hospitals, as this would require a far more complicated process. For now, it is important to create a simplified web user interface, preserve functionalities used by researchers, and get rid of ParaView.

3.2.1 Functional Requirements

All required features for the web application are listed below.

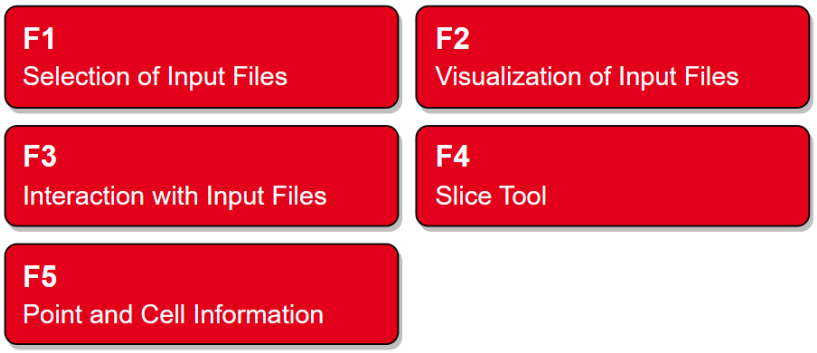

Figure 3.2: The overview of functional requirements

Shortcut: F1

Title: The Selection of the Input Files

Description: The user is able to load the VTI files. Such a file represents a 3D image of a certain human organ (typically a heart with an aorta). Because these files are the output of 4D flow MRI, they also include several data arrays (i.e., anatomy, three velocity components, and velocity magnitude through the different depths, including the denoised version). It is possible to switch files and their data arrays on the fly. All the provided data is anonymized.

Shortcut: F2

Title: The Visualization of the Input Files

Description: The web application is able to visualize the VTI files and their data arrays. The user is able to switch the representation of the visualized VTI file. It is possible to control the player (and see the blood flow in the heart, for example) if there is a group of more VTI files.

Shortcut: F3

Title: The Interaction with the Input Files

Description: The user is able to manipulate and interact with the VTI data displayed in the web application by using the set of available support tools. It is possible to zoom in or out, translate, and rotate the data along the three basic (x, y, and z) axes.

Shortcut: F4

Title: Slice Tool

Description: The user is able to slice the VTI data with a plane and see the cut. It is possible to change the position and orientation of the cutting plane.

Shortcut: F5

Title: Point and Cell Information

Description: The web application is able to display information about the point or cell if the user clicks on it. For a point, it shows an identifier, coordinates, and the values of point data arrays. For a cell, it shows an identifier and the values of cell data arrays. The display of this information is exclusive (either the point, cell, or none of them).

3.2.2 Non-Functional Requirements

All required properties for the web application are listed below.

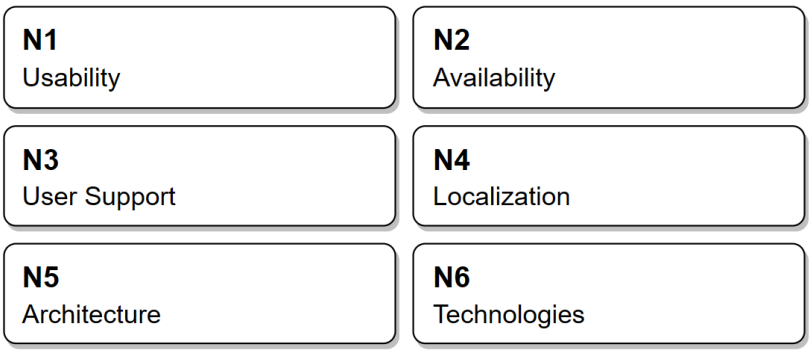

Figure 3.3: The overview of non-functional requirements

Shortcut: N1

Title: Usability

Description: The application is going to be used by researchers. Another expected user is also a radiologist (i.e., a doctor specialized in diagnosis of diseases with the help of imaging methods such as MRI). Therefore, it is critical to include only simple and relevant features. Users do not have to understand programming or technology.

Shortcut: N2

Title: Availability

Description: The web application is publicly available through the web browser. It is not required to be compatible with different devices (desktop computers only), but compatibility with modern web browsers (especially

with Google Chrome and Microsoft Edge) should be satisfied. It also runs continuously, even if not necessary (when research is not in progress).

Shortcut: N3

Title: User Support

Description: All the icons are accompanied by descriptions that help users understand their meaning and consequences of their usage. If it is necessary, the information icons should be used. The user documentation is multilingual and available directly in the web application.

Shortcut: N4

Title: Localization

Description: The web application is available in English by default, but the user can switch to Czech if necessary.

Shortcut: N5

Title: Architecture

Description: The web application is modularized and containerized. It follows various design principles and patterns. Therefore, it is easily extensible, deployable, and maintainable.

Shortcut: N6

Title: Technologies

Description: The web application should be built on modern web technologies. The IKEM plans to create another visual analytics applications in the future. Therefore, it is required to choose appropriate technologies and determine the technical direction.

3.2.3 Use Cases

The requirements are mapped to the use cases as follows:

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12	UC13	UC14	UC15
F1	+	+	+												
F2				+	+	+									
F3							+	+	+						
F4										+	+	+			
F5													+	+	+

Figure 3.4: Mapping requirements to use cases

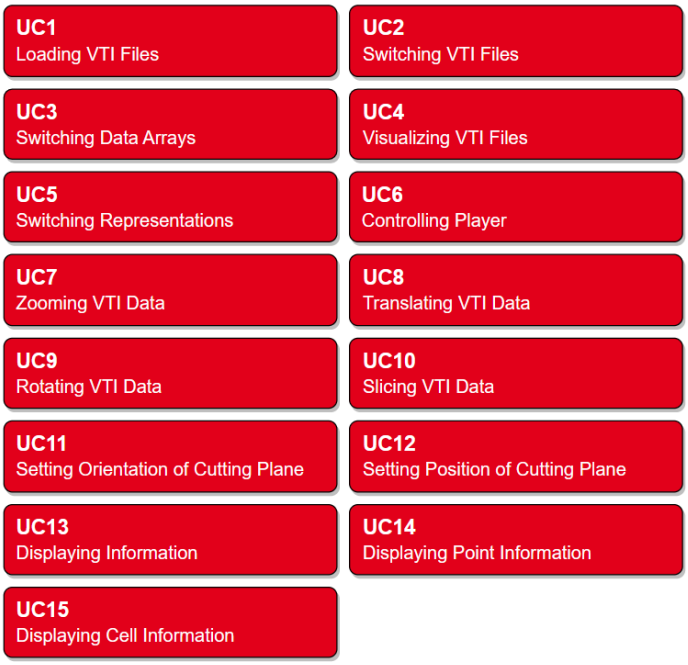

Figure 3.5: The overview of use cases

Shortcut: UC1

Title: Loading VTI Files

Description: The user expects to browse the local computer and load VTI files locally. Additionally, he or she can paste a URL (that is, "Uniform Resource Locator") and load a VTI file remotely.

Scenario: The web application shows a dialog for uploading VTI files that contains two separate sections. The first section includes an icon (*local computer*) and a file input for choosing VTI files from the local computer. After clicking the file input, the web application shows the GUI (that is, "Graphical User Interface") for browsing the file system of the user's computer. When the user is finished with browsing and confirms the selection, the local VTI files are automatically loaded. The second section includes an icon (*remote server*) and a text field for pasting the URL, along with a button for loading a remote VTI file. After pasting the URL into the text field and clicking on the load button, the remote VTI file is loaded over the network. There is also a button to cancel the dialog in the bottom-right corner.

Shortcut: UC2

Title: Switching VTI Files

Description: The user expects to switch between different VTI files after they are uploaded into the web application.

Scenario: The web application shows a select with items. Each item contains the file name of the uploaded VTI file. The user can select only one of these items. After selecting the specific item, the respective VTI file is visualized. If there are many items in the select, a side scrollbar is available.

Shortcut: UC3

Title: Switching Data Arrays

Description: The user expects to switch between different data arrays of the particular VTI file after it is uploaded into the web application.

Scenario: The web application shows a select with items. Each item contains the name of the particular data array. The user can select only one of these items. After selecting the specific item, the respective data array is visualized. If there are many items in the select, a side scrollbar is available.

Shortcut: UC4

Title: Visualizing VTI Files

Description: The user expects the web application to visualize the particular VTI file after it is uploaded or selected. If the user uploads multiple VTI files, then the first one is visualized when the upload is complete.

Scenario: The web application shows the visualization of the respective VTI file with the help of a particular visualization library. The visualization is centered and focused on the camera by default. It can be changed by manipulation or by selecting another VTI file, data array, or representation.

Shortcut: UC5

Title: Switching Representations

Description: The user expects to switch between different representations of the particular VTI file after it is uploaded into the web application.

Scenario: The web application shows a select with items. Each item contains the name of the particular representation. Available representations are *Points*, *Slice*, *Surface*, *Surface with Edges*, and *Wireframe*. The user can select only one of these items. After selecting the specific item, the respective representation is applied and visualized. If there are many items in the select, a side scrollbar is available.

Shortcut: UC6

Title: Controlling the Player

Description: The user expects to play or pause the player and skip to the previous or next VTI file. It is possible to control the player only if there are multiple VTI files uploaded into the web application. If there is only one, the player options are not available.

Scenario: The web application shows multiple icons (*backward*, *play*, *pause*, and *forward*) next to each other. By clicking on the *backward* icon, the previous VTI file is visualized. By clicking on the *forward* icon, the next VTI file is visualized. By clicking on the *play* icon, the player goes through the VTI files and displays them one after another. Each VTI file is shown with a duration of `0.25` seconds. By clicking on the *pause* icon, the player is stopped.

Shortcut: UC7

Title: Zooming the VTI Data

Description: The user expects to zoom in or out on the respective VTI data shown in the web application.

Scenario: The web application shows a section with the "Zoom" title and two buttons (*plus* and *minus*) for zooming in or out. Next to the title, there is an information icon with a hint to use a shortcut for faster work. Under the buttons, a slider for setting the power of zooming is possible to adjust.

Shortcut: UC8

Title: Translating the VTI Data

Description: The user expects to translate the respective VTI data shown in the web application in the direction of three basic axes (x, y, and z).

Scenario: The web application shows a section with the "Translation" title and six *arrow* buttons (each pair for each axis) for moving the data along the specific axis. Next to the title, there is an information icon with a hint to use a shortcut for faster work. Under the buttons, a slider for setting the step of translation is possible to adjust.

Shortcut: UC9

Title: Rotating the VTI Data

Description: The user expects to rotate the respective VTI data shown in the web application around three basic axes (x, y, and z).

Scenario: The web application shows a section with the "Rotation" title and six *rounded-arrow* buttons (each pair for each axis) for rotating the data around the specific axis. Next to the title, there is an information icon with a hint to use a shortcut for faster work. Under the buttons, a slider for setting the angle of rotation is possible to adjust.

Shortcut: UC10

Title: Slicing the VTI Data

Description: The user expects to slice the respective VTI data shown in the web application with a plane.

Scenario: The web application shows a section with the "Slice" title when the *Slice* representation is selected. The slice is also applied to the respective VTI data and visualized with the help of a particular visualization library.

Shortcut: UC11

Title: Setting the Orientation of a Cutting Plane

Description: The user expects to set the orientation of the cutting plane. Available orientations are *XY*, *YZ*, and *XZ*.

Scenario: The web application shows a select with items. Each item contains the name of the particular orientation (*XY*, *YZ*, or *XZ*). The user can select only one of these items. After selecting the specific item, the slice in that respective orientation is applied and visualized. Default orientation is *XY*.

Shortcut: UC12

Title: Setting the Position of a Cutting Plane

Description: The user expects to set the position of the cutting plane in the direction of the remaining axis (e.g., the z-axis for *XY* orientation).

Scenario: The web application shows a slider with the current position of the cutting plane. The slider ranges within the data extent in the direction of the remaining axis. Default position is the minimum value of the data extent in that direction. While moving the slider, the position of the cutting plane changes accordingly.

Shortcut: UC13

Title: Displaying Information

Description: The user expects to choose whether the point or cell information is shown while clicking on the specific primitive in the VTI data.

Scenario: The web application shows a group of three buttons (*off*, *points*, and *cells*). The user can select one of them if he or she wants to show (or hide) specific information about the primitives (i.e., points or cells) after clicking on them. Only one of these buttons is selected at a given time.

Shortcut: UC14

Title: Displaying Point Information

Description: The user expects to see the point information (i.e., an identifier, coordinates, and the values of point data arrays) while clicking on the specific points in the VTI data.

Scenario: The web application shows a dialog with the "Point Information" title. The dialog contains all relevant information about the selected point, which is an identifier, coordinates (x, y, and z), and the values of point data arrays (if they are available in the visualized VTI file). It appears centered on the bottom part of the screen. The dialog can be hidden by clicking anywhere else or by clicking on the *off* or *cells* buttons.

Shortcut: UC15

Title: Displaying Cell Information

Description: The user expects to see the cell information (i.e., an identifier and the values of cell data arrays) while clicking on the specific cells in the VTI data.

Scenario: The web application shows a dialog with the "Cell Information" title. The dialog contains all relevant information about the selected cell, which is an identifier and the values of cell data arrays (if they are available in the visualized VTI file). It appears centered on the bottom part of the screen. The dialog can be hidden by clicking anywhere else or by clicking on the *off* or *points* buttons.

3.2.4 Use Case Diagram

The only actor in the diagram is labeled as a user. Primarily, the user is a researcher, but it can also be a radiologist. Both of them have the same capabilities in the web application. No distinction is made between these roles in the following diagram.

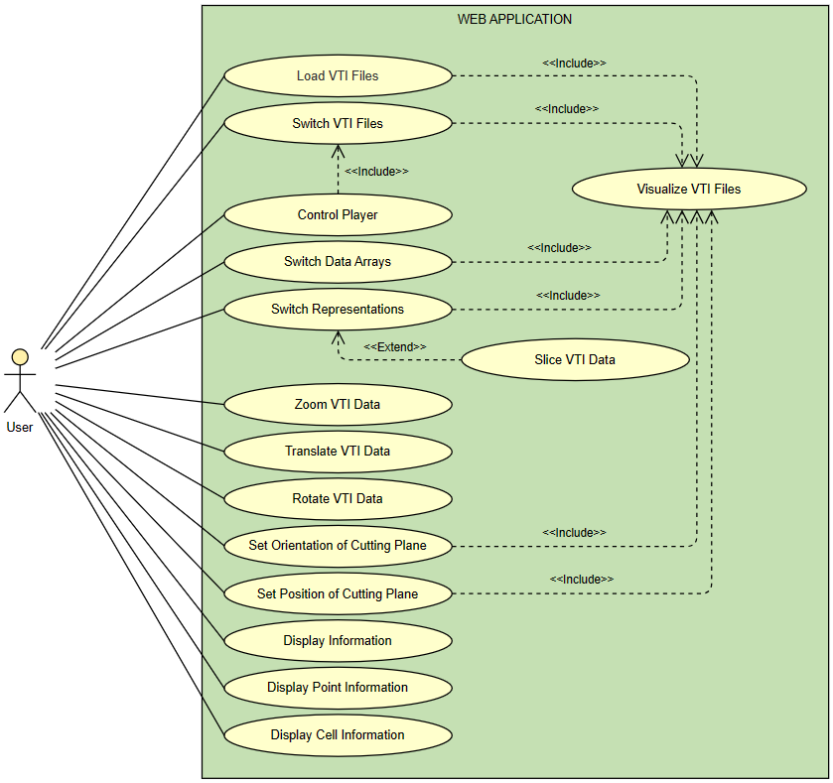

Figure 3.6: Use case diagram

3.3 Similar Solutions

Before discussing technical details and building a new web application, it is important to analyze all the existing similar solutions and check how much they cover the user's requirements and needs. Afterwards, the decision is made. All of the following web applications are based on the ParaViewWeb framework and available on NPM (that is, "Node Package Manager"). They were developed by Kitware, Inc.

3.3.1 Visualizer

The Visualizer application provides a user interface to ParaView accessible in the web browser. In fact, Visualizer is a part of ParaView and can be run with the help of its built-in Python environment. It is very similar to ParaView and, therefore, most suitable to the user's requirements and needs.

The user interface is simplified but still rather complex and overwhelming, containing unnecessary features and parameters hidden in the left column. The provided capabilities (such as slice, player, data array select, or representation select) are similar to those in ParaView. However, Visualizer is rather a flawed and old web application. Icons have no tooltips, and sometimes they are not intuitive. For applying changes (such as a different position of the cutting plane), it is necessary to click on the specific icon. Moreover, the player is not working properly. There are no support tools for transformations, and point or cell information is also missing.

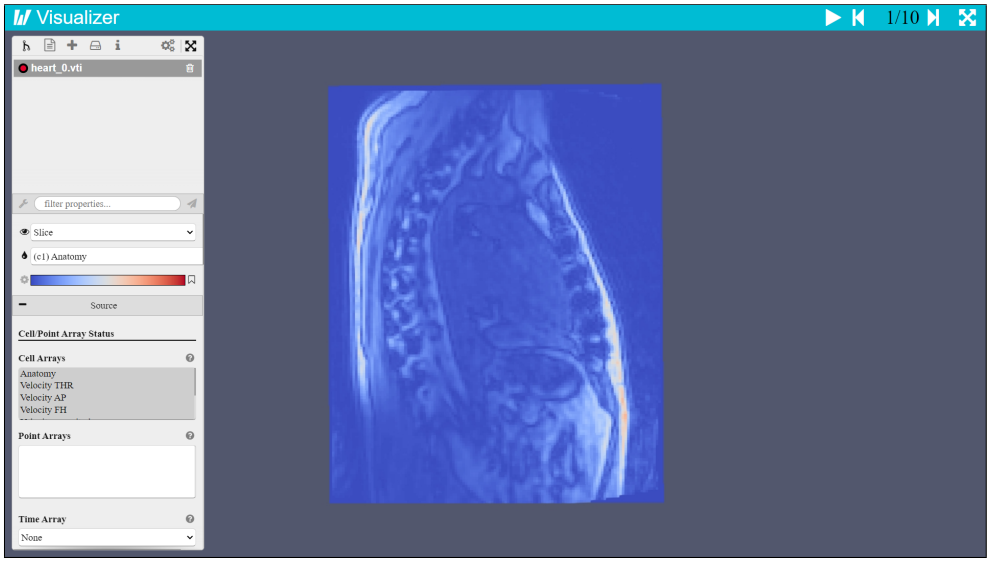

Figure 3.7: Visualization in Visualizer

3.3.2 LightViz

The LightViz application is a visualization tool similar to Visualizer. Likewise, it provides a simplified interactive user interface. Besides the visualization, LightViz offers the mostly used tools from ParaView (e.g., player, slice, multi-slice, volume rendering, etc.). These tools are configurable and can be added or removed as the user requires. Therefore, LightViz is sufficient in terms of complexity. However, it is nowadays a flawed, obsolete, and old web application with three years of no maintenance. Additionally, the developers of LightViz recommend not using this application in production [21].

3.3.3 ArcticViewer

The ArcticViewer application is a standalone scientific visualization tool that serves as a data viewer for large datasets. Apart from Visualizer and LightViz, ArcticViewer does not require ParaView as a backend. Again, it has a simplified user interface for visualization purposes that is accessible in the web browser. Slice, zoom, and pan manipulations are also available. On the other hand, ArcticViewer is a deprecated application that is no longer supported.

3.3.4 Decision

Of course, there are more similar solutions available on the internet. Worth mentioning is the GitHub profile `KitwareMedical` that keeps over 100 repositories with various medical tools and utilities. In summary, all similar solutions found fulfilled only a subset of the users' requirements. Their user interfaces are simplified, but they still have more features than desired. Additionally, some of them cannot even be launched due to their obsolescence, high error rate, and lack of maintenance. Overall, similar solutions are not suitable for the researchers at the IKEM. Therefore, it is necessary to develop a new web application tailored to our users' needs and requirements.

3.4 Technical Ways

For the time being, previous and desired states were described in this chapter, including all problems and requirements that users have. The web application is certainly going to be based on a visualization library and available in the web browser. Therefore, it is necessary to analyze the technical options and make a decision. All of the following technical ways are ordered from the most challenging to the most convenient in terms of realization.

3.4.1 WebGL

WebGL is a cross-browser and cross-platform API (that is, "Application Programming Interface") for creating 3D graphics directly in the web browser. It

is a low-level library based on OpenGL and GLSL (that is, "OpenGL Shading Language") that runs in the HTML5 (that is, "HyperText Markup Language") `<canvas>` element. Therefore, WebGL can be easily integrated into the web application and work along with HTML5, CSS3 (that is, "Cascading Style Sheets"), and JavaScript. [22]

3.4.2 Three.js

Three.js is a 3D library that streamlines the creation of 3D graphics in the web browser. It leverages WebGL and simplifies work with this low-level API. Therefore, it is possible to create stunning and powerful web applications with 3D graphics in a high-level way. The basic building blocks of Three.js are the renderer, camera, scene, meshes, geometries, materials, textures, and lights. Moreover, the library offers plenty of built-in geometry primitives. On the other hand, Three.js is a modern framework, and its compatibility with the old browsers is not guaranteed. However, it has comprehensive and well-arranged documentation. Similarly to WebGL, it can also be easily integrated with HTML5, CSS3, and JavaScript. [23]

3.4.3 ParaViewWeb

ParaViewWeb is an open-source JavaScript library for creating web applications that use scientific visualizations and run in the web browser. The framework leverages Three.js for handling 3D graphics, but it is migrating to VTK.js (that is, "Visualization Toolkit"). It is possible to use ParaView or VTK as a backend for data processing and rendering. Communication between the client and the server is provided by WebSocket. [24]

The library offers several modules regarding data handling, interaction, rendering, and UI (that is, "User Interface"). It provides many React and visualization components (such as 1D or 2D histograms, parallel coordinates, various kinds of editors, etc.) that support a user in interaction, exploration, focusing on details, customizing, and modifying the visualization. [24]

Working with ParaViewWeb requires deep knowledge of web development and modern JavaScript (i.e., Webpack, Babel, ES6, etc.). In the end, a developer only picks features and capabilities that are desired. Nowadays, there are many web applications powered by ParaViewWeb, such as Visualizer, LightViz, ArcticViewer, and so forth. [24]

3.4.4 Trame

Trame is an open-source Python library for building interactive, fashionable, and powerful visualization applications. The framework works along with VTK (i.e., a library for visualizing scientific data) and Vuetify (i.e., a framework for building user interfaces). It is built on ParaViewWeb, but the overall

underlying complexity is hidden. Moreover, minimal knowledge of web development and modern technologies is required. This approach makes the development of web applications much more simple and fast. [25]

3.4.5 Decision

All of the previous technical ways can be used to solve the users' problems and fulfill their requirements. However, each of them is more or less appropriate. WebGL is considered inappropriate because it is a low-level API, and even the simplest things take a lot of code to write. Such an inconvenience is successfully solved by Three.js through its simplified and high-level work with objects such as renderer, camera, scene, and so forth. On the other hand, the support of VTI files is problematic in this library. Taking into account the users' requirements, Three.js is not appropriate either. ParaViewWeb is considered appropriate because it is the complete web framework for building web applications with scientific visualizations. However, it still takes knowledge, money, and time to make such a web application. Trame is much more modern, and the building of web applications is simplified and accelerated. On the other hand, Trame is a truly high-level framework, and it may not be easy to get into low levels to change some configuration parameters, for example. According to *N6*, the client is looking for modern web technologies and planning to build another visual analytics applications in the future. Additionally, both Trame and ParaView are based on VTK. All these technologies are developed by Kitware. Actually, it is advantageous, and the features of ParaView should be feasible in Trame. In summary, Trame is considered the best option. Trame, VTK, and Vuetify will be described in more detail later.

Chapter 4

Design

Based on the previous analysis, the final solution is proposed in this chapter. It was designed with the help of personas as the archetypes of our users. Therefore, personas are described at first. Then, the chapter continues with wireframes drawn immediately after collecting all requirements. Afterwards, the final design of the web application is presented using the prototype that was created with knowledge of the final technologies before the implementation phase. Finally, the next chapter concentrates on the implementation details of the MVP (that is, "Minimum Viable Product").

4.1 MRI Viewer

The proposed solution is a web application called "MRI Viewer". Basically, it is a viewer of scientific VTI files with data coming from MRI machines. The tool is intended for the analysis of 4D flow MRI medical data. Its greatest benefits are availability through the web browser, complexity reduction, a simplified work process, better user support, localization, and modern technologies described later. *MRI Viewer* is oriented especially towards the researchers.

4.2 Personas

Personas are fictional characters that represent specific groups of users. Each persona is a group of behaviors, goals, characteristics, needs, and opinions. It is helpful for designers, as they can understand their users and look at the design through the eyes of a specific persona.

There are three types of personas: typical user (labeled as A), occasional user (B), and negative user (C). Persona A is the most important user (i.e., a researcher). The web application is designed especially for him or her. These users are also the main testers in the usability testing discussed later. Persona B is a secondary user (i.e., a doctor). It is not expected that this user will

use the web application on a regular basis, but he or she should be able to handle it. Persona C is an antipersona. The web application is not designed for these users, as they will probably not understand and use it. All three personas are described in the following subchapters. Each of them is based on the information gathered from interviews, calls, and observations.

4.2.1 Persona A

Type: Typical user

Name: Elise Wilson

Age: 30

Gender: Female

Hobbies: Reading, traveling, meeting with new people, painting with a focus on details, writing papers and research publications

Description: Elise is a research scientist at a specialized research facility. She studied very hard and achieved a PhD title. She gets up every day at 7 a.m. Her working hours are from 9 a.m. to 5 p.m. Her work focuses mostly on research. She is not afraid of working with modern software tools, applications, and programming languages. In her opinion, they make our lives easier and more effective. There is no problem with learning new things and approaches for her. Otherwise, she is kind, helpful, and curious, sometimes shy. From time to time, she visits conferences on her favorite topic: medical care technologies. However, her medical background is rather limited. On the other hand, she has vast experience with image processing and 3D visualizations. Being punctual and focused on details are some of her strengths.

4.2.2 Persona B

Type: Occasional user

Name: Walter Fritz

Age: 45

Gender: Male

Hobbies: Helping and talking to other people, cooking healthy food and drink, running, drinking tea, self-education, training new doctors, tennis

Description: Walter is a doctor with a specialization in radiology. He is rather introverted by nature. In his childhood, he avoided parties, fun, and computer games. Instead, he was focused on his future job as a radiologist. He studied really hard to become a doctor at a specialized medical facility. That is why he is so successful and smart today. He has managed to have effective time management. He avoids useless things and rather works effectively as much as possible. If he works with complex and difficult software, he loses patience very soon. On the other hand, he welcomes simple, effective, and well-organized support tools that make his work go faster.

4.2.3 Persona C

Type: Negative user

Name: Roy Dennis

Age: 68

Gender: Male

Hobbies: Gardening, watching TV, reading newspapers, spending time with his family, walking, meeting with his friends in the pub or cafe, petting his dog, discussing his problems and politics

Description: Roy is a retired newsagent's worker. He used to have an immutable routine in his life. Therefore, he does not like learning new things. He wants to have his work done as soon as possible. Working with mobile phones and computers makes him nervous. He is afraid of doctors and the machines they use because he does not know how they work. Being in a hospital and having examinations makes him feel anxious.

4.3 Wireframes

After finishing the first round of requirements analysis, six frontend screens were painted using the Balsamiq tool. Their objective was to gain a more comprehensive understanding of the client's requirements. Wireframes cover all use cases and show only the initial design because they were created without knowledge of final technologies. All of them were discussed and approved by the client. Of course, there are other requirements (not captured on screens) that were added in later phases. Only a few significant wireframes are included in the following text. The attached CD contains the rest of them.

The following figure depicts the startup options that are shown first while approaching the web application. It covers the *UC1* use case. After choosing VTI files, there was another screen where the user could change the selection of data arrays to be loaded. However, this was classified as a redundant step and skipped in the final prototype.

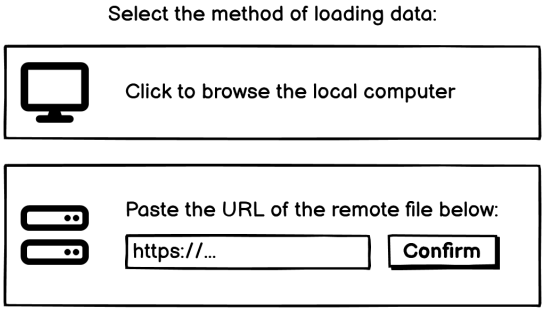

Figure 4.1: Loading options

The web application's working environment is shown in the wireframe below. It covers most use cases (i.e., *UC2–10* and *UC13*). All tools are organized evenly around the visualization. The emphasis is on a minimalistic user interface with the least possible number of functionalities. Every tool has a help icon for assistance that explains its usage. Additionally, there is a section containing a filter for reducing the noise of the visualized data. However, denoised data was added as a new data array to the VTI files of the researchers, and this section was no longer required. Therefore, it was skipped in the final prototype.

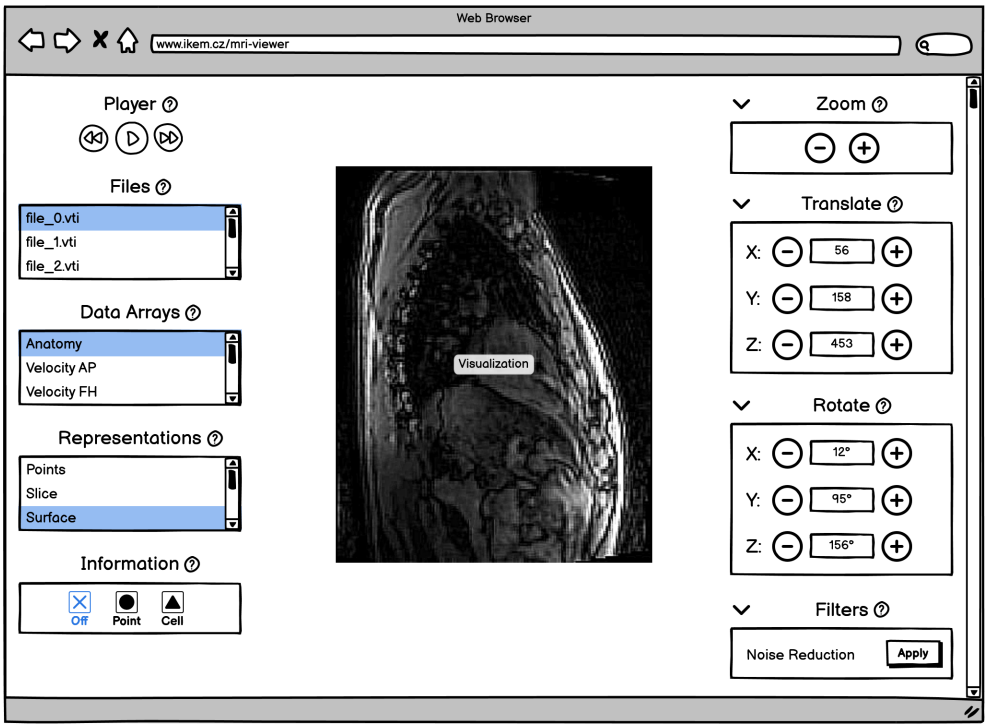

Figure 4.2: Working environment

In the following wireframe, information about the particular cell (i.e., the *UC15* use case) is shown while hovering the mouse over it. It was later modified to show the information only after clicking the respective cell. The reason was that the user was overwhelmed by the information of the surrounding irrelevant cells each time he or she moved a mouse.

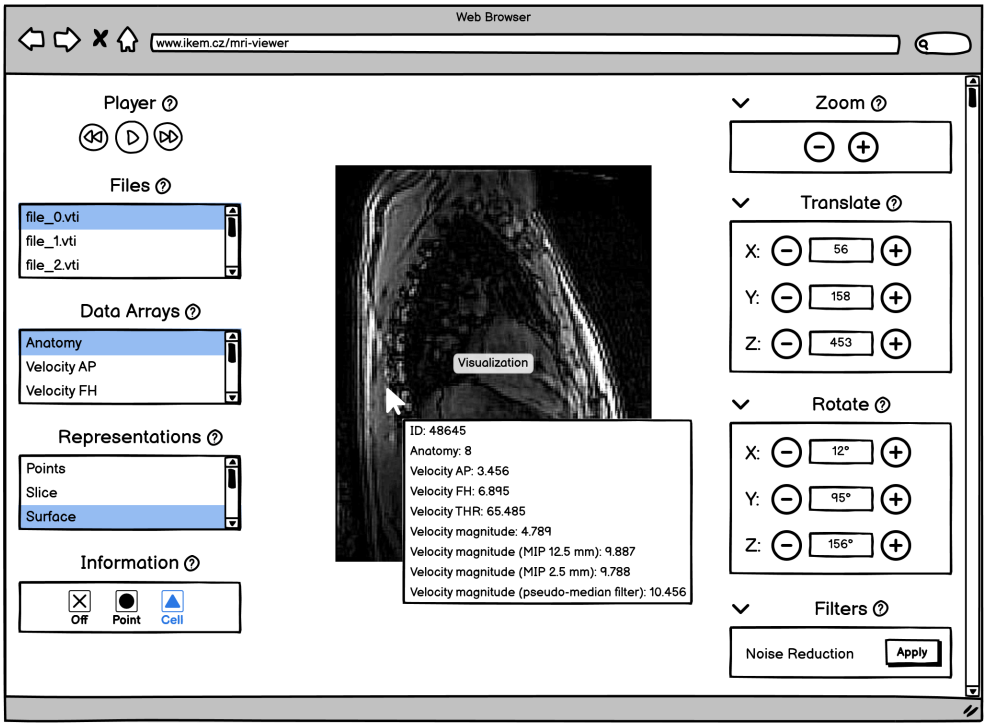

Figure 4.3: Cell information

4.4 Prototype

The final prototype was built with the help of the Axure tool. In contrast to wireframes, the prototype is clickable. It contains colors that match those of the IKEM (i.e., red and white). Not all features were included, as Axure and other prototyping tools are simply not capable of implementing them (for example, the rotation of a 3D visualization). However, the presented user interface is based on Vuetify, which was used for the implementation. There are minimal differences between the final prototype and implementation, even though many other features were added later (such as axes information, axes widget, scalar bar, resetting the view, different themes, etc.). Based on the final prototype, use case scenarios were created and further used for usability testing. The prototype was discussed with and approved by the client. The full version is available on the attached CD.

The first part of the prototype is a dialog for uploading files that shows up at startup. It provides two ways of loading data (locally or remotely), accompanied by an introductory text on limitations. These limitations are checked later in the code, keeping the user posted if they are violated.

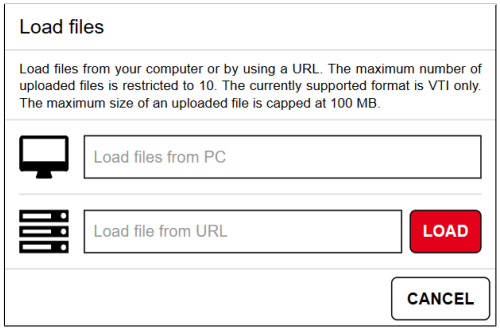

Figure 4.4: Dialog for uploading files

The second part of the prototype is the working environment that appears after uploading files. It consists of the upper bar, left column, and main content for visualizing the uploaded files.

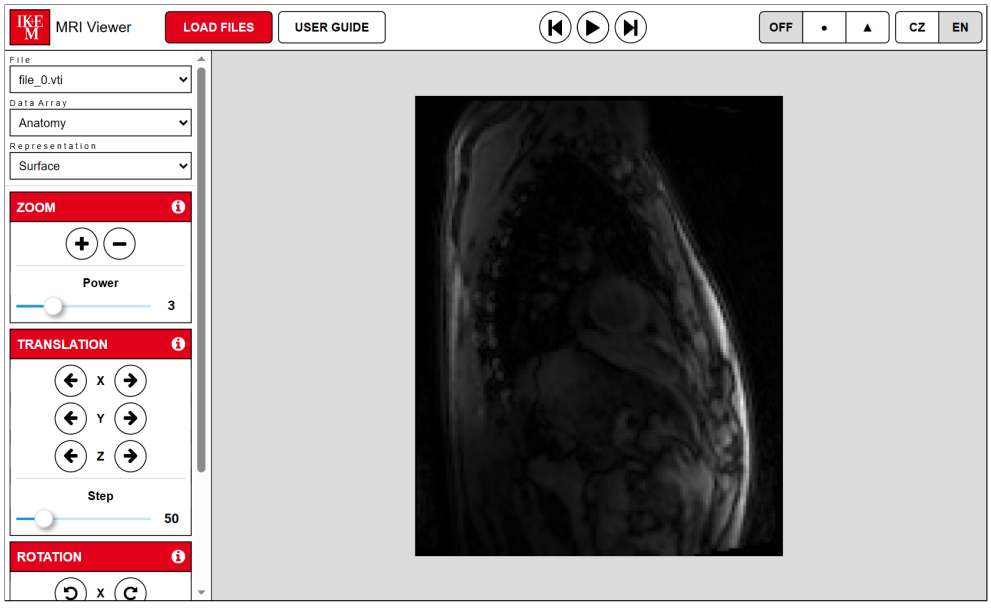

Figure 4.5: Working environment

Eventually, select components were used instead of menus as they save space, which is a difference from wireframes. The only disadvantage is one extra click for the user. Selects hold files, data arrays, and representations. More select components can be easily added in the future.

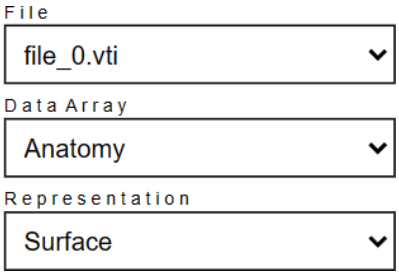

Figure 4.6: Selects

The upper bar contains the IKEM logo with a link to their website, a button to open the dialog for uploading files, and a multilingual user guide. Furthermore, there is also a player, point and cell information control, and a language switch.

Figure 4.7: Upper bar

The following figure shows all the tools available in the left column. Transformation tools (i.e., zoom, translation, and rotation) contain information icons that recommend using specific shortcuts for faster work. Thus, they are only supportive and intended for novice users. These tools also contain sliders for customizing their factors (e.g., the angle of rotation).

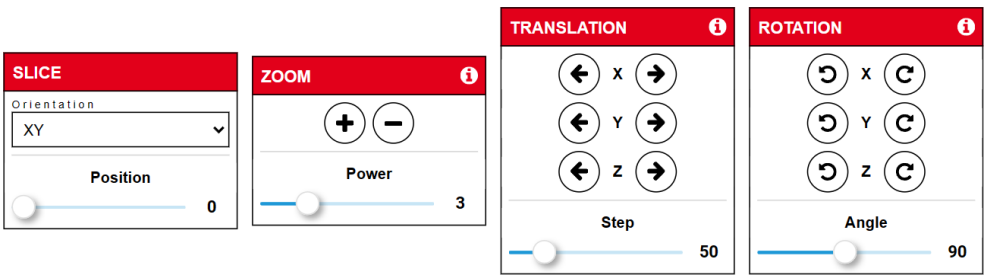

Figure 4.8: Tools

Chapter 5

Implementation

This chapter provides a description of the web application's implementation. The final implementation is based on the client's requirements and the prototype, which were both discussed in the previous chapters. The content of this chapter is further supplemented with auxiliary diagrams. Testing will be discussed in the following chapter.

5.1 Technologies

MRI Viewer relies on three modern frameworks and libraries – *Trame*, *VTK* (including *VTK.js*), and *Vuetify*. The author of these technologies (except *Vuetify*) is Kitware, Inc. This is also the creator of the ParaView tool, which was previously used by the client. These mentioned technologies are described in the following subchapters, as they will be needed later in the text.

5.1.1 Trame

Trame is an open-source, stateful web framework for developing interactive visualization applications in plain Python. It was initially released in 2021 and has been rapidly evolving since then [26]. [25]

The framework is built upon ParaView, ParaViewWeb, VTK, VTK.js, Vue.js, HTML5, CSS3, and other underlying technologies, but their complexity is completely hidden. There is no need for deep knowledge of web development to build a Trame application. A Python developer simply writes *.py* scripts and can mainly focus on data analysis, manipulation, and visualization. With this approach, it is possible to build visual analytics applications easily and faster than before. [25]

Other tools (such as Matplotlib, Pandas, or NumPy) can be integrated with the help of `pip` (PyPI) or Conda. After all, Trame itself is implemented as a Python package, too. It is also an integration framework, which means that the building of a web application is just a matter of orchestrating several

Python packages together. Any missing functionality can be easily added. However, it is still the best practice to install all the dependencies within the Python virtual environment so that the global space is not cluttered. [25]

5.1.1.1 MVVM Pattern

The client-server architecture of Trame is based on the MVVM (that is, "Model-View-ViewModel") pattern. It is necessary for a developer to define each of the three components to make a working application. [25]

Model represents the business logic of the application. It consists of functions, methods, and classes that define behaviors together. They can modify the shared state (ViewModel) and bind events to functions managed by the controller. With the help of `@change` decorators, it is also possible to react to the changes in the shared state. [25]

View represents the user interface of the application. It is based on Vuetify, which is a Vue.js component library, but the whole UI is declared in Python. View converts the shared state into the UI, where the user can modify it (e.g., by clicking a button or moving a slider). It is always up-to-date after making changes to the shared state. [25]

ViewModel represents the shared state of the application. It is a dictionary for important serializable data to be presented in the View. Its purpose is to synchronize the client with the server and bind the UI with the business logic. The shared state can be modified internally (from within the business logic) or externally (from the user interface). Variables from the shared state can be bound to particular UI elements. [25]

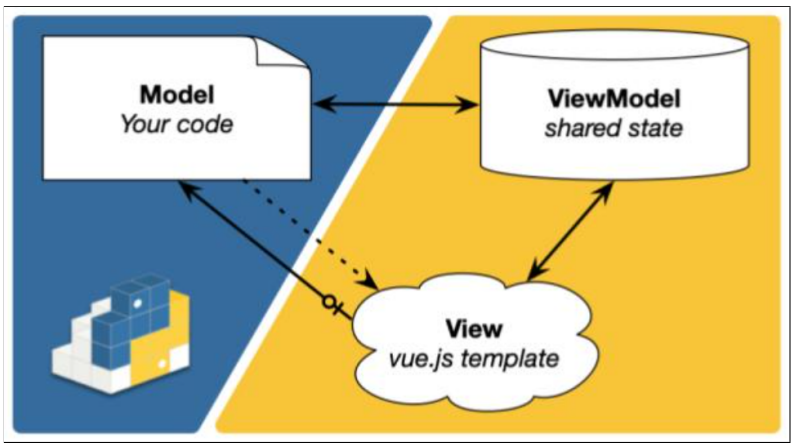

Figure 5.1: The MVVM pattern of Trame [27]

5.1.1.2 Cookiecutter

Cookiecutter is a CLI (that is, "Command Line Interface") utility for the instant creation of a new project from a ready-made project template. These templates are called "cookiecutters", and they are used to generate mostly Python projects. On the other hand, there are many cookiecutters available on the internet today. It is possible to generate a wide range of projects that include but are not limited to Python, Django, Flask, or even Go and C projects. Using the Cookiecutter tool at the beginning of the development phase can significantly speed up the whole process. [28]

The authors of Trame also created their own cookiecutter for generating Trame projects. Their cookiecutter creates a sample Trame project, which is a web application for visualizing a simple cone. Moreover, there is a code infrastructure for various kinds of deployment. Therefore, Trame applications can easily be:

- deployed in the cloud with the help of Docker;
- distributed as desktop applications (for Windows, Linux, and MacOS);
- uploaded to PyPI (that is, "Python Package Index"); or
- run in JupyterLab. [29]

The generation process of Trame applications is very simple. At the beginning, it is necessary to install Cookiecutter (using `pip`, for instance). Afterwards, a developer runs the `cookiecutter` command with a reference to the Trame's cookiecutter. Then, the developer answers a set of basic questions (e.g., project name, type, author, description, license, etc.). Finally, the basic code template is generated. [29]

5.1.2 VTK

VTK (that is, "Visualization Toolkit") is an open-source library for visualizing, manipulating, and interacting with scientific and medical data. It is a backward-compatible, object-oriented, and truly complex system capable of converting data (e.g., *.vti*, *.vtp*, *.dcm*, *.stl*, *.ply*, *.obj* file formats) into graphical representations. Besides, VTK is also able to modify the data, play animations, show different widgets, and much more. Fortunately, VTK is already a part of the Trame framework and can be integrated into the Python virtual environment using `pip`. The best way to learn such a comprehensive tool is by inspecting examples and studying the VTK's user guide. [19]

The library is composed of the compiled C++ core and an interpreted (e.g., Python) wrapper for working with this core. The basic flow is to read (or generate) some data, render it, and let the user interact with it. There are two main VTK components that are used to achieve it:

- **Visualization pipeline** reads (or creates) data, processes it, and writes it to a file or passes it to the rendering engine to visualize.
- **Rendering engine** creates the visual representation of the passed data and renders the result into a window. [19]

VTK accepts data of various types (e.g., *vtkImageData*, *vtkPolyData*, *vtkStructuredGrid*, etc.). These data contain *points* (as single geometric spots in space) and *cells* (as single topological groups of points in space). Moreover, they may include attribute data (such as *scalars*, *vectors*, *normals*, *tensors*, etc.) associated with them. To be able to render the data using VTK, several objects are required:

- **Mapper** holds a reference to the raw data and converts it to a visual representation.
- **Actor** connects to the *mapper* and represents the data in the scene. It uses the *property* object to control the appearance of the data.
- **Renderer** creates a scene and puts *actors*, *camera*, and *lights* into it. It is responsible for rendering.
- **RenderWindow** connects the operating system with VTK. It opens a platform-specific window and manages the display process. Several *renderers* can be contained.
- **RenderWindowInteractor** listens and processes the events with the help of a Command/Observer design pattern to provide corresponding features, such as rotating, panning, or zooming. [19]

5.1.2.1 VTK.js

VTK.js is the implementation of VTK in vanilla JavaScript ES6. The goal of this library is to make the visualization capabilities of VTK available in the web browser, where it leverages WebGL. However, VTK.js is still in development. Therefore, it should not be considered equivalent to VTK. [30]

5.1.2.2 Visualization Example

The following code snippet shows the visualization of a cone using VTK. It was written in the Python programming language. The image below the code displays the output.

```
# Create polygonal cone
cone_source = vtkConeSource()

# Map polygonal data (geometry) to graphic primitives
cone_mapper = vtkPolyDataMapper()
cone_mapper.SetInputConnection(cone_source.GetOutputPort())

# Represent object (geometry and properties) in rendered scene
cone_actor = vtkActor()
cone_actor.SetMapper(cone_mapper)

# Create helper for accessing named colors
colors = vtkNamedColors()

# Set color of actor to be white
cone_actor.GetProperty().SetColor(colors.GetColor3d("White"))

# Create renderer to control rendering process for actors
renderer = vtkRenderer()

# Create window for renderers to draw their images into
render_window = vtkRenderWindow()
render_window.AddRenderer(renderer)

# Create interactor to provide interaction mechanism for events
render_window_interactor = vtkRenderWindowInteractor()
render_window_interactor.SetRenderWindow(render_window)
render_window_interactor.Initialize()

# Add actor to renderer
renderer.AddActor(cone_actor)

# Set camera to see every actor
renderer.ResetCamera()

# Command local renderers to render their image
render_window.Render()

# Start event loop
render_window_interactor.Start()
```

Code 5.1: The basic visualization in VTK

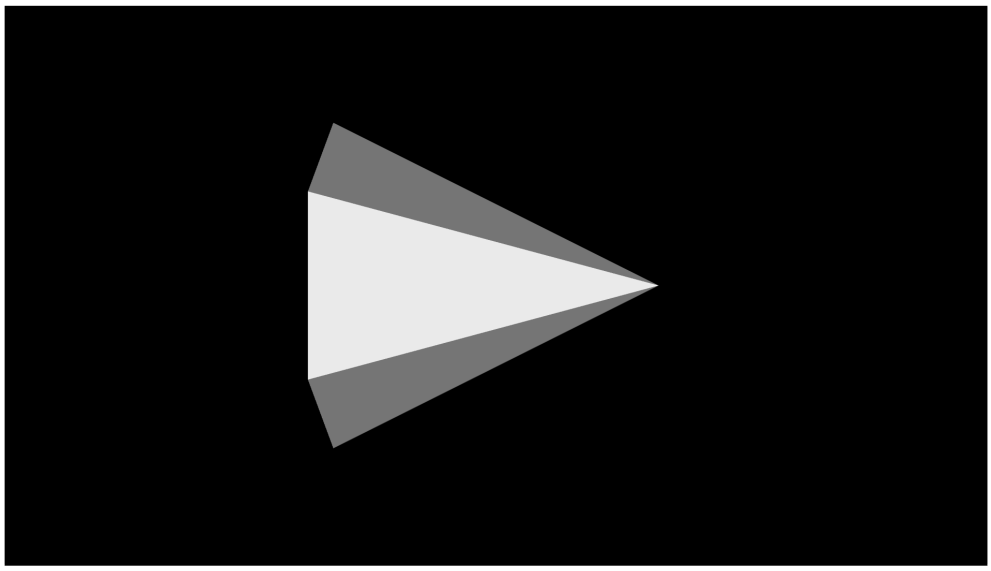

Figure 5.2: The output of the visualization example

5.1.3 Vuetify

Vue.js is a popular JavaScript framework for building user interfaces. Similarly to React and Angular, the user interface here is composed of components. A component is a small, reusable module of HTML, CSS, and logic that represents some part of the user interface. Vuetify is an open-source library of customizable, ready-made, and responsive Vue.js components (for instance, *autocomplete*, *card*, *carousel*, *dialog*, *file input*, *snackbar*, *stepper*, *tooltip*, and many more). Thanks to the comprehensive documentation, clear API, and large community, it is easy to learn, and no design skills are needed. Developers can use Vuetify to build UI and UX (that is, "User eXperience") simply and effortlessly, saving their time. [31]

5.1.3.1 Usage in Trame

The user interface of a Trame application is based on Vuetify. However, Trame uses its own version of Vuetify that is completely rewritten in Python. Consequently, there are syntax differences between both versions. Vuetify is used in two groups of elements that Trame provides – UI and widgets. [25]

UI is a group of layout components. Layouts are top-level components that define the organization and structure of the whole UI. Additionally, they define regions in the UI where widgets can be placed. The user interface of the Trame application is built on a single layout component, which might be:

- `VAppLayout` (i.e., a blank fullscreen);
- `SinglePageLayout` (i.e., `VAppLayout` with icon, title, toolbar, content, and footer); or
- `SinglePageWithDrawerLayout` (i.e., `SinglePageLayout` with a column on the left side). [25]

Widgets is a group of basic elements (including but not limited to Vuetify components) that are used to fill the predefined layout. [25]

5.2 Development

The development of *MRI Viewer* took four months in total. It was written and debugged in Visual Studio Code as the primary IDE (that is, "Integrated Development Environment"). The project is stored in a public repository on GitHub and contains three Git branches: *main*, *dev*, and *deploy*. Naturally, the code is accompanied by Python docstrings and comments.

Technologies used for the development (i.e., Trame, VTK, and Vuetify) were completely new and unknown to the author of this thesis. Moreover, Trame is a very modern and rapidly evolving technology in the field of web-based visualization. The lack of experience, in combination with the originality of Trame, caused time delays and issues while developing.

The method of learning by examples (available on the internet) was the most effective one, even though many examples were obsolete, because the syntax has slightly changed with every new version of Trame. Moreover, the official Trame documentation (including the code documentation) remains unfinished, which caused issues (for instance, while deploying). Although Kitware (as the author of Trame) offers support, educational courses, and training opportunities for their users, it is unfeasible from an academic perspective as these services are very expensive (even with student discounts included).

However, Kitware stores projects (including Trame) on GitHub, where users can raise questions in GitHub issues and discussions. In this way, it is possible to get in touch with Kitware developers and solve all the problems together. Additionally, ChatGPT has been used in practice to provide hints on how to implement certain features in VTK. Although the code produced by this LLM (that is, "Large Language Model") was mostly incorrect, it provided specific functions and methods from the VTK's large API that could be explored further. Nonetheless, it was very easy to create a web application using Trame, VTK, and Vuetify, but very hard to develop a practical tool (as *MRI Viewer*) with no prior knowledge of these technologies.

5.3 Web Application

MRI Viewer is a Python package that can be launched as a web (or desktop) application. Additionally, it is also possible to execute it in JupyterLab. The application has been containerized, and therefore, it is easily deployable using Docker. *MRI Viewer* has already been deployed and integrated with the help of FNSPE (that is, "Faculty of Nuclear Sciences and Physical Engineering") CTU (that is, "Czech Technical University").

It leverages Trame, VTK, and Vuetify. Trame provides the basic infrastructure for building the web application. VTK powers visualizations, and Vuetify simplifies the process of building the user interface. *MRI Viewer* was built upon the MVVM pattern with the help of OOP (that is, "Object-Oriented Programming") principles. Based on a client-server model, the client (front end) is used for interacting, and the server (back end) processes requests. The implementation is described in the following subchapters, and the user guide is available in the attachments (appendix B).

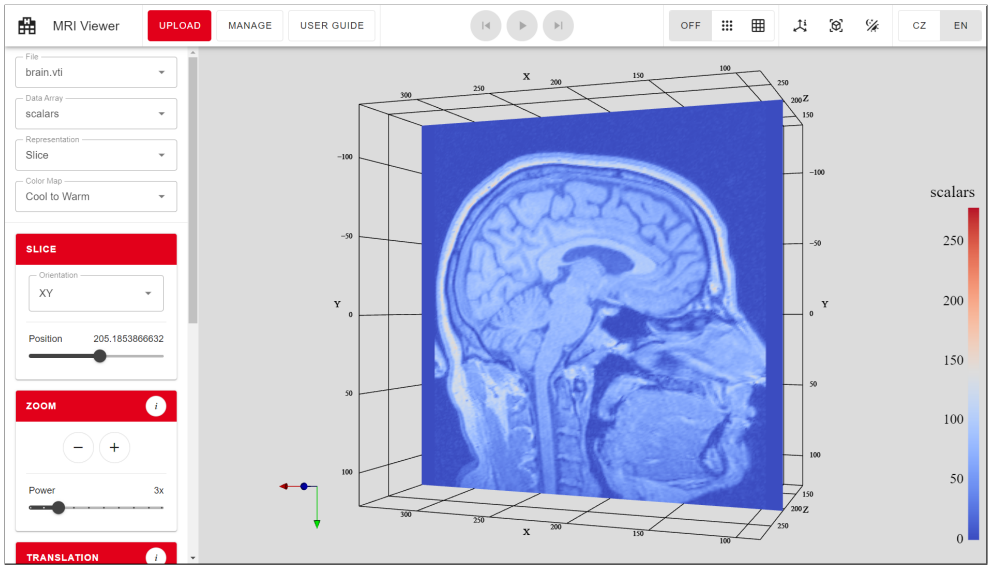

Figure 5.3: The human brain visualized in *MRI Viewer*

5.3.1 Engine

`MRIViewerApp` is a core class of *MRI Viewer*. It holds the server instance and controls the state (ViewModel) of the web application. Moreover, the class has access to the user interface (View) that is built during the initialization time with the `build_ui` method. This method provides a reference to the `SinglePageWithDrawerLayout` as a top-level element.

GUI elements and widgets are linked directly to the state via the `v_model` attribute. `MRIViewerApp` defines listeners to the state variables using the `@change` decorators so that the web application can react to changes from the user interface. Another way of connecting the user interface with the logic (Model) is by using a controller, which is a simple container for methods. Moreover, it serves as a mediator (which is also one of the behavioral design patterns) that helps reduce dependencies between classes by forcing them to use the controller's methods.

In fact, `MRIViewerApp` serves as a high-level facade that uses the concept of managers to divide and conquer the problem. It delegates the work to lower-level specialized classes (called "managers" or "subfacades") that solve smaller problems. This architecture helps to follow the single responsibility principle and prevents `MRIViewerApp` from becoming a god object.

5.3.2 File Manager

The responsibility of the file manager is the management of VTI files. It organizes files into groups. Files within the same group have the same properties (such as extent, origin, spacing, and data arrays). Moreover, groups save parameters set while working with these files (such as current representation, color map, camera view, slice orientation and position, etc.). Therefore, groups automatically remember and restore work in progress when the user switches between files. Besides, the file manager also handles uploading, validating, and deleting files.

5.3.3 Language Manager

The responsibility of the language manager is the management of languages (currently Czech and English only). All languages are represented as dictionaries sharing the same interface. The user can switch between these languages in the web application. Additionally, the language manager provides different user guides based on the selected language.

5.3.4 VTK Manager

The responsibility of the VTK manager is the management of VTK objects. It uses the custom `VTKCreator` class for creating tailored VTK objects (such as renderer, render window, render window interactor, mappers, actors, etc.). Each file group has its own set of VTK objects. Of course, many VTK objects are shared across file groups (such as the render window, because there is only one window for the whole time). VTK manager offers several features, as described in the following subchapters.

5.3.4.1 Interaction

MRI Viewer provides support tools for user interaction and manipulating the visualized data. In fact, the user does not manipulate the data themselves, but the camera while zooming, translating, and rotating. The data remains static for the entire time. For zooming, the manager uses the `Zoom` method of `vtkCamera`. For translating, the manager sets the new camera position and focal point with the help of the `SetPosition` and `SetFocalPoint` methods. For rotation, the manager executes the `Translate` and `Rotate` methods of `vtkTransform` that are applied to the camera. Simply put, the camera is relocated from its original position to the center of the data. Then, the camera rotates around the specific axis. Finally, it is relocated using the same (but negative) translation as before, which moves it to a new position.

5.3.4.2 Picking

Speaking of picking points and cells, it is necessary to synchronize the server camera with the client camera after every user interaction. While clicking the particular point or cell on the client, a new event object emerges containing the position in display coordinates. This event position is sent to the server, where it is used as the parameter of the `Pick` method executed by `vtkCellPicker` on the respective `vtkRenderer`. Basically, `vtkCellPicker` shoots a ray from the specified position into the scene and returns the information about the first actor it hits (for instance, the particular point or cell identificator). Then, the process of retrieving information about a hit point or cell is trivial. Moreover, the affected primitive is highlighted. Picking would be impossible without client-server camera synchronization. Otherwise, the ray would be cast to a different rendered image and return the wrong information.

5.3.4.3 Rendering

MRI Viewer uses `vtkXMLImageDataReader` as a VTI file reader. After reading the particular VTI file, the reader is connected to the `vtkDataSetMapper` object together with the `vtkLookupTable`. The lookup table contains colors gained by the rasterization of the specific `vtkColorTransferFunction`, which is defined by the specific color map (e.g., grayscale). Therefore, scalars saved in the VTI file can be mapped to colors with the help of the lookup table. Afterwards, the mapper is linked to the `vtkActor` that is put into the scene. Finally, the scene is rerendered, and the client is synchronized with the server so that the user can see the most recent image.

5.3.4.4 Slicing

Slicing is based on the `vtkExtractVOI` filter that selects the VOI (that is, "Volume Of Interest") from the specific VTI data. First, the filter is linked to

the data gained by the VTI file reader. Then, it follows the same rendering procedure as for other objects. The filter is linked to the `vtkDataSetMapper` object together with the lookup table. After that, it is connected to the `vtkActor` that is visualized in the scene. In addition, the VOI must be selected using the `SetVOI` method in the range of all three axes. Slicing is done by setting this range to the VTI data extent and narrowing the range of one specific axis to a single unit. Therefore, a 2D image of the VTI data can be rendered and positioned across the specific axis.

5.3.5 Architecture

The following class diagram depicts the architecture of the web application. It contains each of the classes and dependencies between them. Due to the complexity, the diagram is simplified and some parts are deliberately omitted.

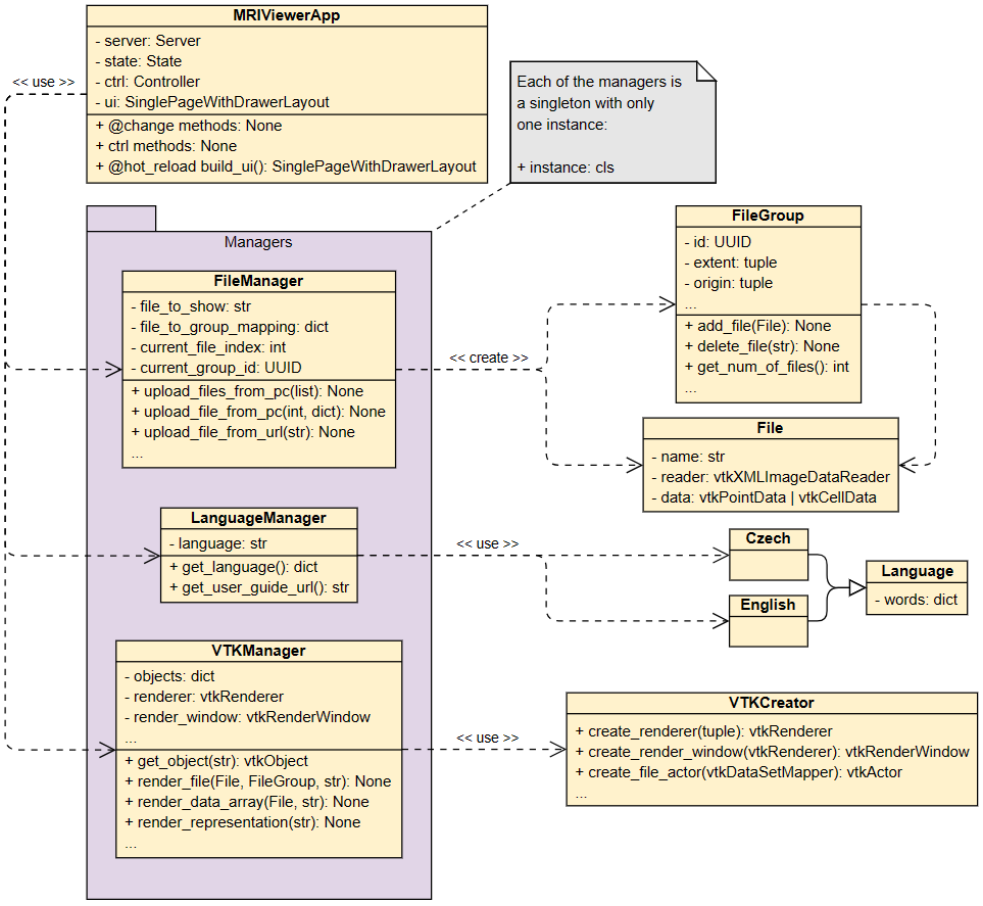

Figure 5.4: The architecture of the web application

5.3.6 User Interface

The user interface of *MRI Viewer* represents the View in the MVVM pattern. It is based on the `SinglePageWithDrawerLayout` layout. The basic building blocks of this layout are *icon*, *title*, *toolbar*, *drawer*, *content*, and *footer*. These blocks are filled with components (i.e., groups of widgets). The following diagram depicts the architecture of the UI in a more clear manner.

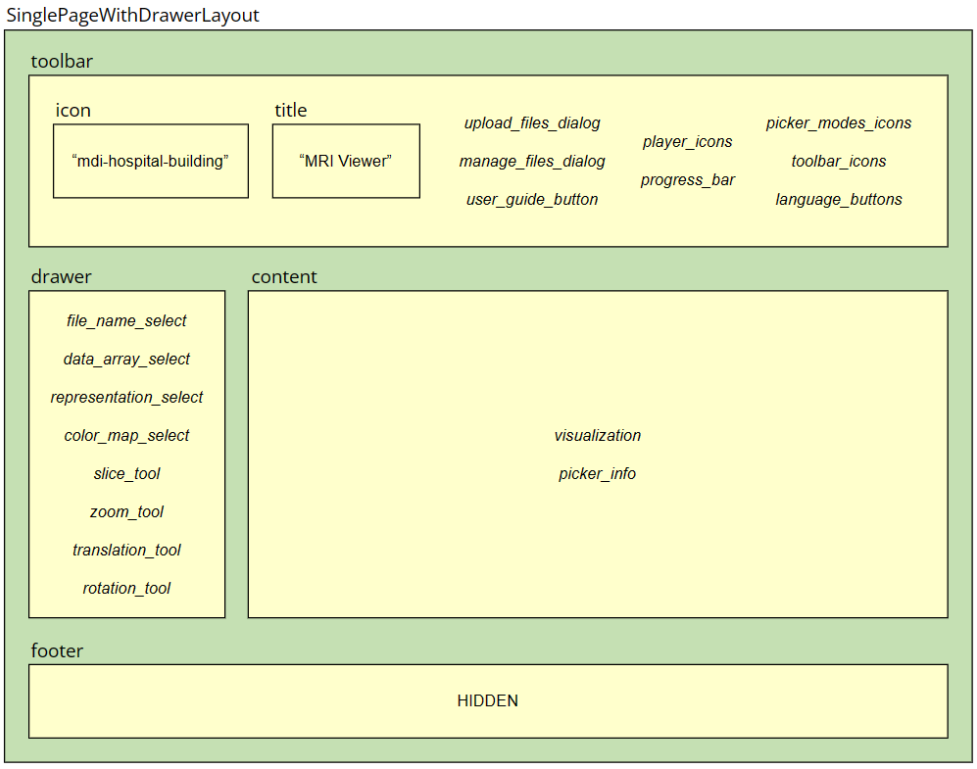

Figure 5.5: The architecture of the user interface

5.3.6.1 Components

A component is a Python function with the respective widgets (e.g., `html`, `vtk`, `vuetify3`, etc.) declared. Components are connected to the shared state with the help of the `v_model` attribute. They can also use the functions or methods stored in the controller for event handling. However, components in *MRI Viewer* do not contain any logic (to keep the MVVM pattern), which is the difference from, for instance, React components. The following list contains each of the components with a little description:

- `upload_files_dialog` consists of a visible button (`vuetify3.VBtn`) that opens the dialog for uploading files (`vuetify3.VDialog`). The dialog is further composed of other elements, such as a file input for uploading files from the computer (`vuetify3.VFileInput`) or a specialized text field for uploading files via URL (`vuetify3.VTextField`).

- `manage_files_dialog` consists of a visible button (`vuetify3.VBtn`) that opens the dialog for managing files (`vuetify3.VDialog`). The dialog contains the list of files where each file (except the visualized one) can be deleted. Furthemore, it includes the confirmation page for the user to confirm the deletion.

- `user_guide_button` is a button (`vuetify3.VBtn`) for opening the user guide of *MRI Viewer* on a specific URL (`html.A`) in a new tab of the web browser.

- `player_icons` consists of three icons (`vuetify3.VIcon`) for controlling the player (i.e., previous file, play/pause, next file). Each of the icons is decorated with the `@tooltip` decorator, which shows a short description for the user after hovering the mouse over the icon.

- `progress_bar` is a component that represents an endless progress indicator (`vuetify3.VProgressLinear`) located under the toolbar. Whenever it is active, it means that the web application is busy processing requests.

- `picker_modes_icons` consists of one button (`vuetify3.VBtn`) for turning off the picking and two icons (`vuetify3.VIcon`) for turning on the point or cell picking. The icons are decorated with a tooltip.

- `toolbar_icons` consists of three icons (`vuetify3.VIcon`) for turning on/off the axes information, resetting the view, and changing the theme. Again, all of them have tooltips.

- `language_buttons` is a group of buttons (`vuetify3.VBtn`) for switching between the Czech and English versions of the web application.

- `file_name_select` is a file selector (`vuetify3.VSelect`) containing all the uploaded files. The user can switch between different files with the help of this component. Afterwards, the selected file is visualized.

- `data_array_select` is a data array selector (`vuetify3.VSelect`) containing all point/cell arrays included in the visualized file. The user can switch between different data arrays with the help of this component.

- `representation_select` is a specialized selector (`vuetify3.VSelect`) for different ways of displaying the selected file. The user can switch between different representations with the help of this component.

- `color_map_select` is a specialized selector (`vuetify3.VSelect`) for different ways of coloring the selected file. The user can switch between grayscale and temperature colors with the help of this component.
- `slice_tool` is a component that represents a tool for controlling the data slicing. It consists of elements, such as a selector (`vuetify3.VSelect`) for different orientations or a slider (`vuetify3.VSlider`) for setting the slice position within the data extent.
- `zoom_tool` is a component that represents a tool for zooming the data. It consists of tooltiped icons (`vuetify3.VIcon`) for controlling the zoom (i.e., zoom in or out). There is also a slider (`vuetify3.VSlider`) for setting the power of the zoom.
- `translation_tool` is a component that represents a tool for translating the data. It consists of tooltiped icons (`vuetify3.VIcon`) for controlling the translation in the directions of all axes. There is also a slider (`vuetify3.VSlider`) for setting the step of the translation.
- `rotation_tool` is a component that represents a tool for rotating the data. It consists of tooltiped icons (`vuetify3.VIcon`) for controlling the rotation around all axes. There is also a slider (`vuetify3.VSlider`) for setting the angle of the rotation.
- `visualization` is a component for performing VTK visualizations and rendering (`vtk.VtkLocalView`). The data is rendered locally on the client side using the hardware resources of the client, which is more performant in terms of FPS (that is, "Frames Per Second"). On the other hand, the data must be transferred from the server, which might cause latency. [32]
- `picker_info` represents a small window (`vuetify3.VCard`) with information (such as an identificator, position, or data array values) appearing while picking points or cells.

All the above components are decorated with the `hot_reload` decorator. Together with the `watchdog` Python package, it performs live updates to the user interface displayed in a web browser when the developer saves a file. Thus, it is not necessary to restart the server each time while tuning the UI.

5.3.7 Design Patterns

Design patterns are commonly known solutions for typical problems in software development. By using them, the code can be more readable, extensible, and maintainable. While developing *MRI Viewer*, several design patterns were used, as described in the following subchapters. There are also code samples (with comments omitted) for each of the patterns.

5.3.7.1 Decorator

Decorator is a structural design pattern that adds new behaviors to wrapped objects if they are placed into wrappers containing these functionalities. Furthermore, it is possible to nest decorators so that the wrapped objects have multiple different behaviors at once.

Python functions can be decorated by adding `@name_of_decorator` above their definitions. It causes the decorated function to be put as an argument into the decorator, which can add new behaviors to it. There are several decorators used in *MRI Viewer*:

- `TrameApp` is a Trame decorator for Trame applications.
- `change` is a Trame decorator for state change. It runs the decorated function every time the particular state variable changes.
- `hot_reload` is a Trame decorator for reloading the decorated function.
- `tooltip` is a custom decorator for adding a simple tooltip to the decorated function, which is used especially for icons.

```
def tooltip(content):
    def wrapper(**kwargs):
        with vuetify3.VTooltip(
            text=kwargs["tooltip"],
            location=kwargs["tooltip_location"],
        ):
            with vuetify3.Template(
                ...
            ):
                content(**kwargs)

    return wrapper

...

@tooltip
def icon(**kwargs):
    ...
```

Code 5.2: The Decorator design pattern in *MRI Viewer*

5.3.7.2 Facade

Facade is a structural design pattern that represents a simplified interface to a complex subsystem (e.g., a framework). The client does not have to deal with the complex code hidden under the subsystem because the facade hides all the details. Instead, the client uses the methods of the interface.

The architecture of *MRI Viewer* is based on classes called "managers". In fact, all of these managers are facades that work with some underlying and complex set of classes. For instance, there is a class `VTKManager` that simplifies the work with the VTK library. By the way, this is the most used facade in the web application.

```
class VTKManager():
    ...

    def render_file(self, ...):
        ...

    def render_data_array(self, ...):
        ...

    def render_representation(self, ...):
        ...

    def set_slice(self, ...):
        ...

    def render(self):
        ...

    def get_picked_point_info(self, ...):
        ...

    def show_picked_point(self, ...):
        ...

    def hide_picked_point(self):
        ...

    ...
```

Code 5.3: The Facade design pattern in *MRI Viewer*

5.3.7.3 Singleton

Singleton is a creational design pattern that represents a class with only one instance available. The constructor of this class is private. Instead, there is a method that returns the existing instance and serves as an access point.

Each of the managers (or facades) in the web application is a singleton because it does not make sense to create more instances. In Python, the singleton class is created with the help of the `__new__` method, as shown in the following code snippet.

```
class FileManager:
    def __new__(cls):
        if not hasattr(cls, "instance"):
            cls.instance = super().__new__(cls)
        return cls.instance

    ...
```

Code 5.4: The Singleton design pattern in *MRI Viewer*

Chapter 6

Testing

The last chapter describes the testing of *MRI Viewer* as a new medical tool designed for researchers from the IKEM. Testing was divided into the testing of features (and writing unit tests for them), compatibility (of devices and browsers), and usability (with end users). This chapter is further followed by a conclusion, a user guide, and test scenarios.

6.1 Tests

There are up to 50 unit tests available for testing the features of *MRI Viewer* (i.e., the controller methods and file, language, and VTK managers). In order to run tests, it is required to install the `pytest` package and execute the command of the same name. However, the test coverage is not 100%. Therefore, the test base is not complete, and more unit tests can be added in the future while building the CI/CD pipeline.

6.2 Test Data

MRI Viewer was tested with up to 25 various VTI files. The IKEM provided 10 VTI files showing the 4D blood flow in the human aorta. These data contains several data arrays, such as anatomy, velocity (in x, y, and z directions), and velocity magnitude (through the different layer thicknesses), including the denoised version. Other VTI files were obtained from the Kitware Data platform or found publicly available on the internet.

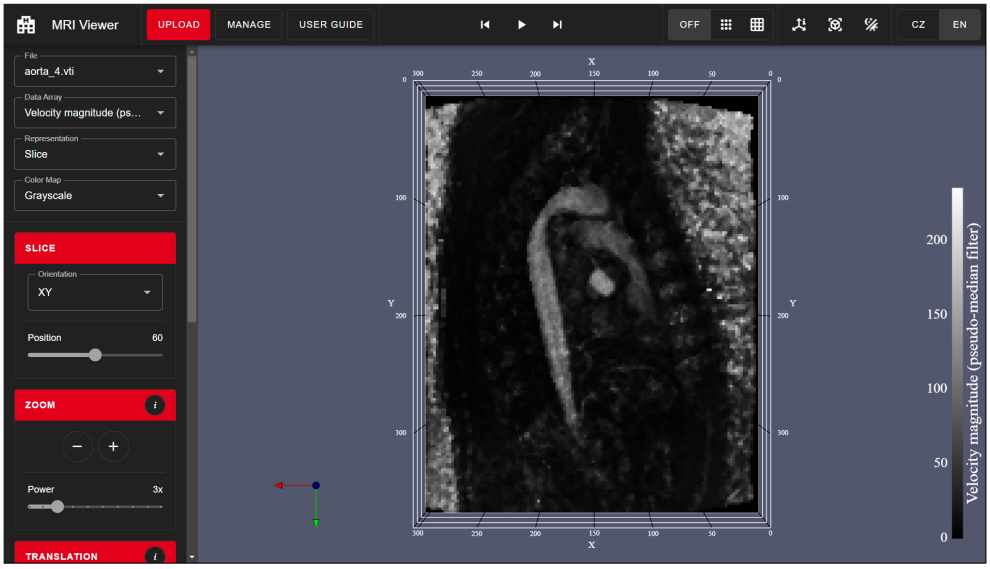

Figure 6.1: The blood flow in the aorta visualized in *MRI Viewer*

6.3 Compatibility

Although it was not required by the client, *MRI Viewer* was tested as a web application, a desktop application, in JupyterLab, and in Docker on the Windows operating system, where it worked with no problems. The web application was also successfully tested with the following versions of web browsers:

- Google Chrome 124;
- Microsoft Edge 124;
- Mozilla Firefox 125; and
- Opera 109.

Regarding different devices, *MRI Viewer* was seamlessly tested on mobile, tablet, laptop, and desktop computers. However, there are no shortcuts or tooltips available on mobile and tablet devices. Moreover, while swiping the sliders of slice, zoom, translation, or rotation tools to the left, it simultaneously closes and hides the left column, where these tools are located. This can be considered a Vuetify flaw. However, these issues were not further addressed, as *MRI Viewer* is primarily used on desktop computers.

6.4 Usability

The purpose of the usability testing was to receive feedback from potential users and modify the web application accordingly. It took place after the completion of the web application. However, there was a problem with the lack of testers. In the end, the usability testing was done with 3 testers of persona A (a researcher) and informally with 2 testers of persona C.

The testing was conducted mostly via Zoom calls (with audio and video enabled). Each test session lasted 30 minutes to an hour. The goal of the usability testing was to let the tester pass all four test scenarios (available in appendix C) that he or she had to go through. These test scenarios covered all features (at the time of the testing) and represented common user activities in the web application.

At the beginning, the moderator introduced himself and started the conversation. The tester shared audio, video, and screen. After that, the moderator introduced the tester to *MRI Viewer* and the usability testing. Then, the tester gradually passed all four test scenarios with the help of the moderator's instructions. At the end, the moderator thanked the tester and said goodbye.

6.4.1 Results

The usability testing was successful. All testers provided huge feedback and ideas so that *MRI Viewer* could be further improved. In this section, there are the results from the testing. These results do not contain findings coming from the testers of persona C, as they will never use the web application.

- All testers provided a comprehensive, in-depth discussion regarding potential *nice2have* features;
- All testers required picking while slicing to measure the blood flow, which was unavailable during the usability testing. This feature was implemented later, together with the scalar bar;
- The layout of the dialog for uploading files was modified and divided into different screens (i.e., options menu, uploading files from PC, and uploading file from URL);
- Selectors items are newly alphabetically sorted (except for data arrays);
- A new feature (i.e., a dialog for managing files) was added;
- A new feature (i.e., a color map selector providing grayscale and temperature color maps) was added;
- A new *nice2have* feature (i.e., slice in an arbitrary orientation) on top of the current slice capabilities was proposed for future development;

- A new *nice2have* feature (i.e., measuring blood flow in the selected region of interest) on top of the current picking capabilities was proposed for future development; and
- A new *nice2have* feature (i.e., storing uploaded VTI files by user account) on top of the current file management capabilities was proposed for future development.

Conclusion

In the theoretical part of this thesis, the reader was briefly introduced to the IKEM, MRI and its principles, ParaView, and two medical file formats, DICOM and VTI. Next, the analysis of previous and desired states, together with the analysis of similar existing solutions, was conducted. Furthermore, the discussion of technical ways to solve the problem was also included.

In the practical part of the thesis, a solution in the form of a web application was proposed based on the previous analyses. The wireframes and the final prototype were elaborated, described, and communicated with the client. Finally, the implementation of the web application was explained, including the testing. Proposals were UI-oriented, and modern technologies, such as Trame or Vuetify, were used.

The main goal of this master's thesis was to develop a web application for visualizing VTI files. This goal was met in the form of a new web application named *MRI Viewer* that complied with all client's requirements and needs. Furthermore, it was containerized and deployed on the servers of FNSPE CTU with the help of Docker, which makes it available and reachable for the researchers at the IKEM. It means that all partial goals (i.e., analysis, design, implementation, and testing) along with optional goals (i.e., deployment) were also successfully met and accomplished.

Nowadays, *MRI Viewer* is fully available for the IKEM. It was built on Trame, which is a new modern technology from 2021. Therefore, the development of *MRI Viewer* was very demanding and time-consuming due to the unfinished documentation of the framework, frequent minor (or major) version changes, and obsolete examples available on the internet. However, *MRI Viewer* is the web application that can be further built upon. The CI/CD pipeline for faster delivery of new versions is currently being worked on. Furthermore, there are certain intentions to build another visual analytics applications from the client's side. This master's thesis, together with *MRI Viewer*, may serve as a guide for future developers (besides the fact that it already simplifies the work of researchers at the IKEM).

In the future, there are plans to extend *MRI Viewer* in 2025, although new features are categorized as *nice2have* by the client or belong to the author's ideas. Some of them cannot even be implemented now as they require the work of Kitware. Their technology is still evolving (such as VTK.js), and some features will only work in the future.

Speaking of *MRI Viewer*, it is still possible to refactor and improve speed and performance (regarding uploading files, picking, or client/server synchronization). There will be a general slice that will enable the user to set a slice in any orientation (not only *XY*, *YZ*, and *XZ*). Other types of representations are planned, such as volume rendering. Additionally, support for the DICOM file format is also intended. Finally, the progress bar should be improved by adding percentages and the current state while uploading files. There are many more new ideas on how to enhance the current version of *MRI Viewer*, but it still needs to be discussed with the client so as not to increase the complexity of the web application and keep it simple.

Bibliography

[1] *IKEM – Institute for Clinical and Experimental Medicine* [online]. IKEM. [cit. 2024-04-02]. Available at: https://www.ikem.cz/en/.

[2] The logo of IKEM. In: *IKEM – Institute for Clinical and Experimental Medicine* [online]. IKEM, 2024. [cit. 2024-04-02]. Available at: https://www.ikem.cz/en/usek-reditele/odbor-pr-a-marketingu/a-2199/.

[3] NICHOLLS, M. Paul Lauterbur and Sir Peter Mansfield for MRI. In: *European Heart Journal* [online]. June 2019, vol. 40, issue 24, pp. 1898-1899. ISSN 1522-9645. [cit. 2024-04-02]. Available at: https://doi.org/10.1093/eurheartj/ehz397.

[4] *MRI – Mayo Clinic* [online]. Mayo Foundation for Medical Education and Research. [cit. 2024-04-02]. Available at: https://www.mayoclinic.org/tests-procedures/mri/about/pac-20384768.

[5] *Magnetic Resonance Imaging (MRI)* [online]. National Institute of Biomedical Imaging and Bioengineering. [cit. 2024-04-02]. Available at: https://www.nibib.nih.gov/science-education/science-topics/magnetic-resonance-imaging-mri.

[6] The MRI Machine. In: *IKEM – Institute for Clinical and Experimental Medicine* [online]. IKEM, 2024. [cit. 2024-04-02]. Available at: https://www.ikem.cz/en/o-nas/zakladni-informace/a-11/.

[7] *Reading and Visualizing Structural MRI Data – Neural Data Science in Python* [online]. Aaron J. Newman. [cit. 2024-04-02]. Available at: https://neuraldatascience.io/8-mri/read_viz.html.

[8] *Why Is an MRI So Expensive at a Hospital?* [online]. South Jersey Radiology Associates. [cit. 2024-04-02]. Available at: https://sjra.com/why-is-an-mri-so-expensive-at-a-hospital/.

[9] SCHILD, H. H. *MRI Made Easy (...Well Almost)* [online]. Berlin: Schering AG, 1990. ISBN 3-921817-41-2. [cit. 2024-04-02]. Available at: https://rads.web.unc.edu/wp-content/uploads/sites/12234/2018/05/Phy-MRI-Made-Easy.pdf.

[10] GROVER, V. P. et. al. Magnetic Resonance Imaging: Principles and Techniques: Lessons for Clinicians. In: *Journal of Clinical and Experimental Hepatology* [online]. September 2015, vol. 5, issue 3, pp. 246-255. ISSN 2213-3453. [cit. 2024-04-02]. Available at: https://doi.org/10.1016/j.jceh.2015.08.001.

[11] Johns Hopkins Medicine. MRI Physics – Magnetic Resonance and Spin Echo Sequences – Johns Hopkins Radiology [video]. *YouTube* [online]. Johns Hopkins Medicine, 2022. [cit. 2024-04-02]. Available at: https://www.youtube.com/watch?v=jLnuPKhKXVM.

[12] *Cardiac Magnetic Resonance Imaging (MRI)* [online]. American Heart Association, Inc. [cit. 2024-04-03]. Available at: https://www.heart.org/en/health-topics/heart-attack/diagnosing-a-heart-attack/magnetic-resonance-imaging-mri.

[13] STANKOVIC, Z., B. D. ALLEN, J. GARCIA, K. B. JARVIS and M. MARKL. 4D Flow Imaging with MRI. In: *Cardiovascular Diagnosis & Therapy* [online]. April 2014, vol. 4, issue 2, pp. 173-192. ISSN 2223-3660. [cit. 2024-04-03]. Available at: https://doi.org/10.3978/j.issn.2223-3652.2014.01.02.

[14] WYMER, D. T., K. P. PATEL, W. F. BURKE and V. K. BHATIA. Phase-Contrast MRI: Physics, Techniques, and Clinical Applications. In: *RadioGraphics* [online]. January 2020, vol. 40, issue 1, pp. 122-140. ISSN 1527-1323. [cit. 2024-04-03]. Available at: https://doi.org/10.1148/rg.2020190039.

[15] *ParaView – Open-source, multi-platform data analysis and visualization application* [online]. Kitware, Inc. [cit. 2024-04-03]. Available at: https://www.paraview.org/.

[16] LAROBINA, M. and L. MURINO. Medical Image File Formats. In: *Journal of Digital Imaging* [online]. April 2014, vol. 27, issue 2, pp. 200-206. ISSN 0897-1889. [cit. 2024-04-04]. Available at: https://doi.org/10.1007/s10278-013-9657-9.

[17] *DICOM* [online]. The Medical Imaging Technology Association. [cit. 2024-04-04]. Available at: https://www.dicomstandard.org/.

[18] *VTI File Extension – What Is It? How to Open a VTI File?* [online]. FILExt. [cit. 2024-04-04]. Available at: https://filext.com/file-extension/VTI.

[19] KITWARE, INC. *The VTK User's Guide* [online]. 11th Edition. Columbia: Kitware, Inc., 2010. ISBN 978-1-930934-23-8. [cit. 2024-03-22]. Available at: https://vtk.org/wp-content/uploads/2021/08/VTKUsersGuide.pdf.

[20] *Get Support From Our Experts – Kitware Europe* [online]. Kitware, Inc. [cit. 2024-04-06]. Available at: https://www.kitware.eu/get-support/.

[21] *GitHub - Kitware/light-viz* [online]. GitHub, Inc. [cit. 2024-04-12]. Available at: https://github.com/kitware/light-viz.

[22] *Getting Started - WebGL Public Wiki* [online]. The Khronos Group, Inc. [cit. 2024-04-11]. Available at: https://www.khronos.org/webgl/wiki/Getting_Started.

[23] *Fundamentals - Three.js Manual* [online]. Three.js. [cit. 2024-04-11]. Available at: https://threejs.org/manual/#en/fundamentals.

[24] *ParaViewWeb* [online]. Kitware, Inc. [cit. 2024-04-11]. Available at: https://kitware.github.io/paraviewweb/docs/.

[25] *Trame* [online]. Kitware, Inc. [cit. 2024-03-14]. Available at: https://kitware.github.io/trame/.

[26] *Releases – Kitware/trame* [online]. GitHub, Inc. [cit. 2024-03-20]. Available at: https://github.com/Kitware/trame/releases.

[27] MVVM Pattern: Model-View-ViewModel. In: *Trame* [online]. Kitware, Inc., 2023. [cit. 2024-03-20]. Available at: https://kitware.github.io/trame/guide/.

[28] *cookiecutter/cookiecutter* [online]. GitHub, Inc. [cit. 2024-03-21]. Available at: https://github.com/cookiecutter/cookiecutter.

[29] *Kitware/trame-cookiecutter* [online]. GitHub, Inc. [cit. 2024-03-21]. Available at: https://github.com/Kitware/trame-cookiecutter.

[30] *Overview – vtk.js* [online]. Kitware, Inc. [cit. 2024-03-22]. Available at: https://kitware.github.io/vtk-js/docs/.

[31] *Vuetify — A Vue Component Framework* [online]. Vuetify. [cit. 2024-03-21]. Available at: https://vuetifyjs.com/en/.

[32] *VTK – Trame* [online]. Kitware, Inc. [cit. 2024-03-26]. Available at: https://kitware.github.io/trame/guide/tutorial/vtk.html.

Appendix A

Acronyms

API Application Programming Interface

CLI Command Line Interface

CMRI Cardiac Magnetic Resonance Imaging

CSS Cascading Style Sheets

CT Computed Tomography

CTU Czech Technical University

DICOM Digital Imaging and Communications in Medicine

FNSPE Faculty of Nuclear Sciences and Physical Engineering

FPS Frames Per Second

GLSL OpenGL Shading Language

GUI Graphical User Interface

HPC High-Performance Computing

HTML HyperText Markup Language

IDE Integrated Development Environment

IKEM Institute for Clinical and Experimental Medicine

LLM Large Language Model

MPI Message Passing Interface

MRI Magnetic Resonance Imaging

MVP Minimum Viable Product

MVVM Model-View-ViewModel

NPM Node Package Manager

OOP Object-Oriented Programming

PC-MRI Phase-Contrast Magnetic Resonance Imaging

PyPI Python Package Index

UI User Interface

URL Uniform Resource Locator

UX User eXperience

VOI Volume Of Interest

VTI Visualization Toolkit Image Data

VTK Visualization Toolkit

XML eXtensible Markup Language

Appendix B

User Guide

The full version of this document is available on the attached CD.

Introduction

Welcome to the user guide for the *MRI Viewer*. Please review the following table of contents and refer to the appropriate section in order to resolve your issue as quickly as possible. If you did not find what you were looking for, proceed to the last chapter and contact us directly via email. We are still developing this user guide, and your feedback is very valuable to us. Thank you, and happy visualizing!

Table of Contents

(skipped)

B.1 MRI Viewer

MRI Viewer is a web application for visualizing *.vti* files. The purpose of the application is to simplify scientific work with data coming from MRI machines. It is a replacement for the ParaView tool in terms of specific workflows and aims to reduce its complexity and enhance its availability. The original client is the IKEM in Prague, Czech Republic.

B.2 Features

MRI Viewer offers several features, as stated in the following chapters.

B.2.1 Upload Files

This feature enables you to upload *.vti* files to the application. You can use this functionality on startup or by clicking the *Upload* button.

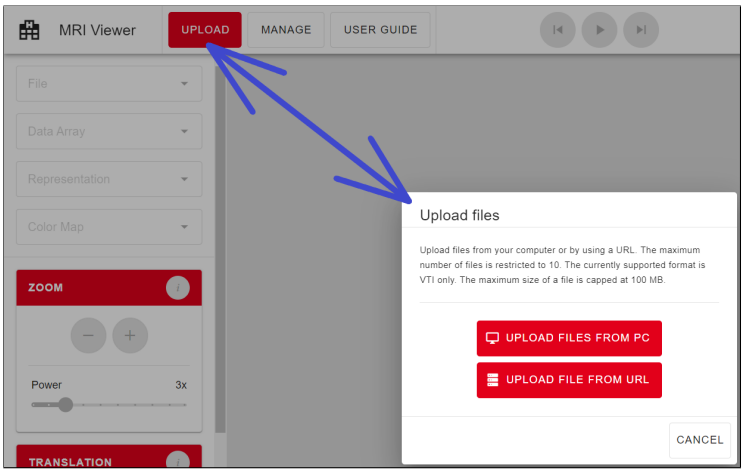

Figure B.1: A dialog for uploading files and a button to open it

B.2.1.1 Limitations

- You can either upload files from your computer or provide a URL.
- You can upload only *.vti* files.
- Each of the uploaded files must not exceed 100 MB.
- If you want to upload data from your computer, you can upload one or more files, but no more than 10.
- If you want to upload files using the URL, you can upload only one file at a time.

B.2.1.2 Error Codes

There is a list of common error codes that can occur while uploading files:

- **WRONG-FILE-EXTENSION**

  You are trying to upload a file with an extension other than *.vti*, which violates the limitation. Please upload *.vti* files only.

- **FILE-IS-TOO-LARGE**

  You are trying to upload a file larger than 100 MB, which violates the limitation. Please compress your data or use smaller files.

- **TOO-MANY-FILES-TO-UPLOAD**

  You are trying to upload more than 10 files, which violates the limitation. Please load your data in small batches (e.g., groups of three files).

- **INVALID-URL**

  You provided an incorrect URL. Please check the URL and make sure there is a *.vti* file on the other side.

- **MISSING-...**

  Error codes starting with "MISSING-" indicates that there are some troubles with reading uploaded *.vti* files, and some parts may be missing. It may also happen that there is no *.vti* file to be uploaded (while uploading via URL). Please check the *.vti* file you are trying to upload, or use another *.vti* file.

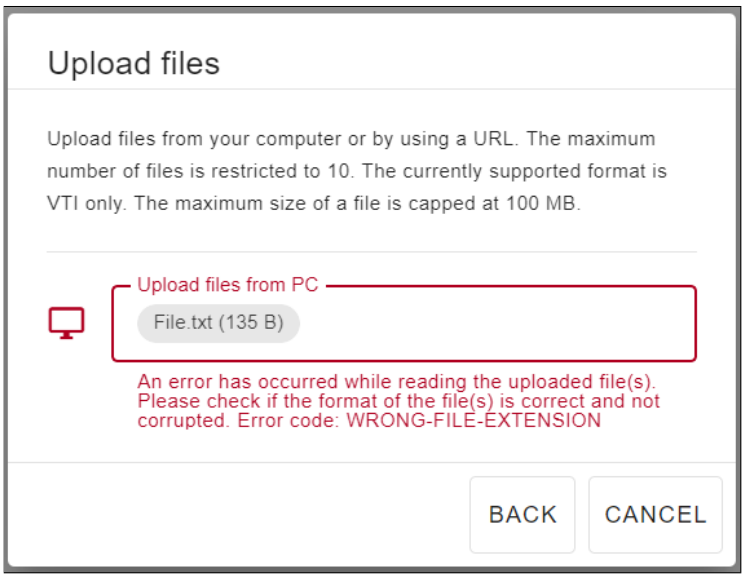

Figure B.2: An example of an error while uploading files

If you encounter an error code that is different from those mentioned above, please contact us and let us know.

B.2.1.3 Recommendations

Please be patient while uploading *.vti* files. This action may take some time (up to one minute or even more), as the data must be properly loaded and processed. It depends not only on the size of your files. Do not upload a large group of files at once. Try to upload data in small groups of files instead.

B.2.2 Manage Files

This feature enables you to delete uploaded *.vti* files. You cannot delete the file that is currently visualized. Therefore, there is always at least one *.vti* file in the web application after the initial upload. After clicking the *Delete* button, another dialog appears to confirm the deletion.

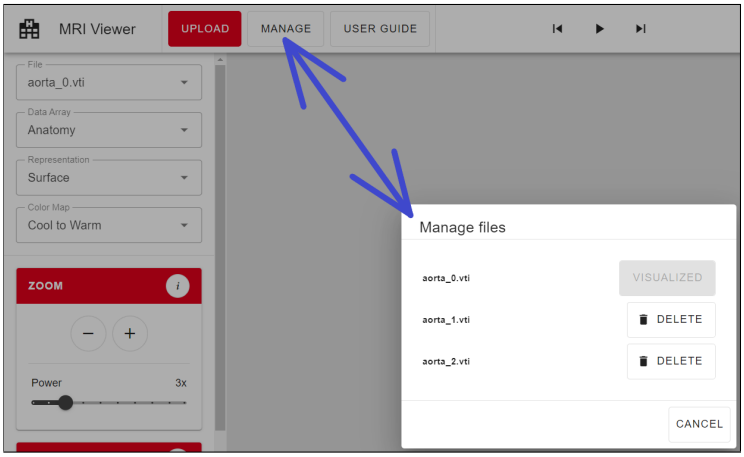

Figure B.3: A dialog for managing files and a button to open it

B.2.3 File

This feature enables you to select one of the uploaded *.vti* files to show. You can use this functionality by clicking the *File* select and choosing the file to visualize. If you upload a group of files, then the first file in this group is selected and visualized. The visualized file is initially temperature-colored.

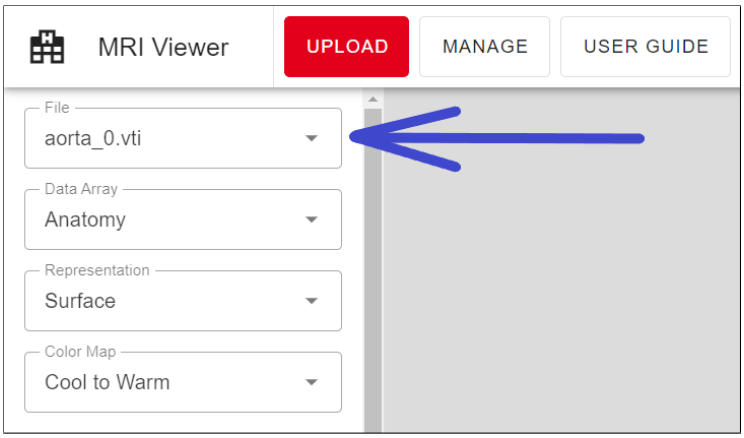

Figure B.4: The list of files to visualize is located in the left column

B.2.3.1 Groups

A file group is automatically created while uploading a file. It always contains at least one file. If you upload 10 different files, then there are implicitly 10 different file groups (each of them contains one file). The main advantage of the file group is that it remembers your work.

If there is already a file group with a very similar file(s) (in terms of data), then the newly uploaded file is added to this group. Groups with more than one file can be played using the player.

B.2.3.2 Memorization

File groups are able to remember the following things: current slice (in all orientations), zoom, translation, rotation, selected data array, representation, and color map. If you switch to another group, then the work in the old group is remembered and the work in progress in the new group is loaded.

B.2.4 Data Array

This feature enables you to select one of the data arrays defined in the selected file. You can use this functionality by clicking the *Data Array* select and choosing the data array to apply. The list contains only point or cell arrays. If there are point and cell arrays defined in a single file simultaneously, then only cell arrays are listed.

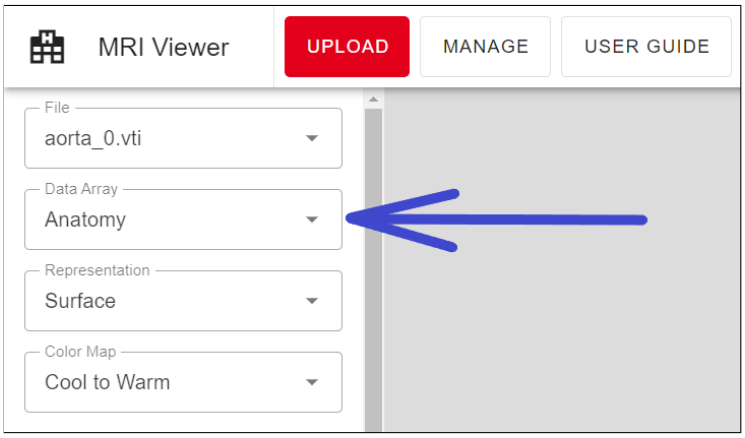

Figure B.5: The list of data arrays to apply is located in the left column

B.2.5 Representation

This feature enables you to select one of the representations of the visualized file. You can use this functionality by clicking the *Representation* select and

choosing the representation to apply. You can represent the visualized file as *points*, *slice*, *surface*, *surface with edges*, or *wireframe*.

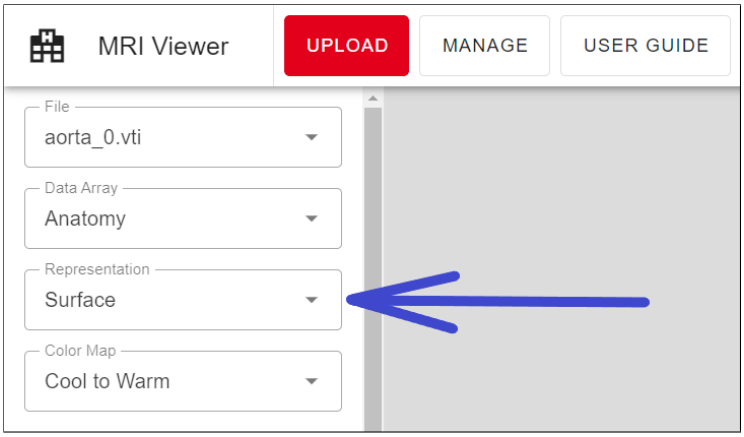

Figure B.6: The list of representations is located in the left column

B.2.6 Color Map

This feature enables you to select one of the color maps of the visualized file. You can use this functionality by clicking the *Color Map* select and choosing the color map. Currently, you can use *grayscale* or *temperature* color maps.

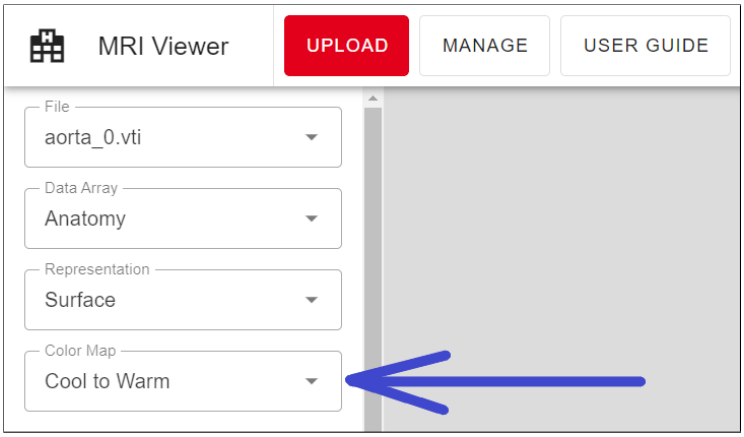

Figure B.7: The list of color maps is located in the left column

B.2.7 Interaction

You can use several available tools to interact with the data, as stated in the following chapters.

B.2.7.1 Slice

This particular tool creates a slice of the data in the pre-selected orientation and position. The list of available orientations contains *XY*, *YZ*, and *XZ*. These are the axes along which the final slice is inserted.

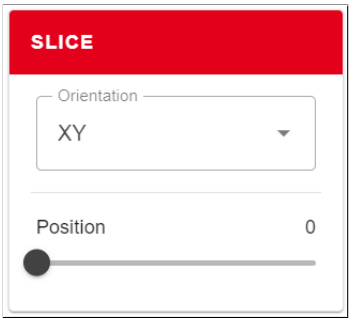

Figure B.8: The slice tool

You can also set the position of the slice in the direction of the remaining axis (e.g., for the *XY* orientation, you set the position within the Z-axis). This position is limited by the data boundaries in that particular direction.

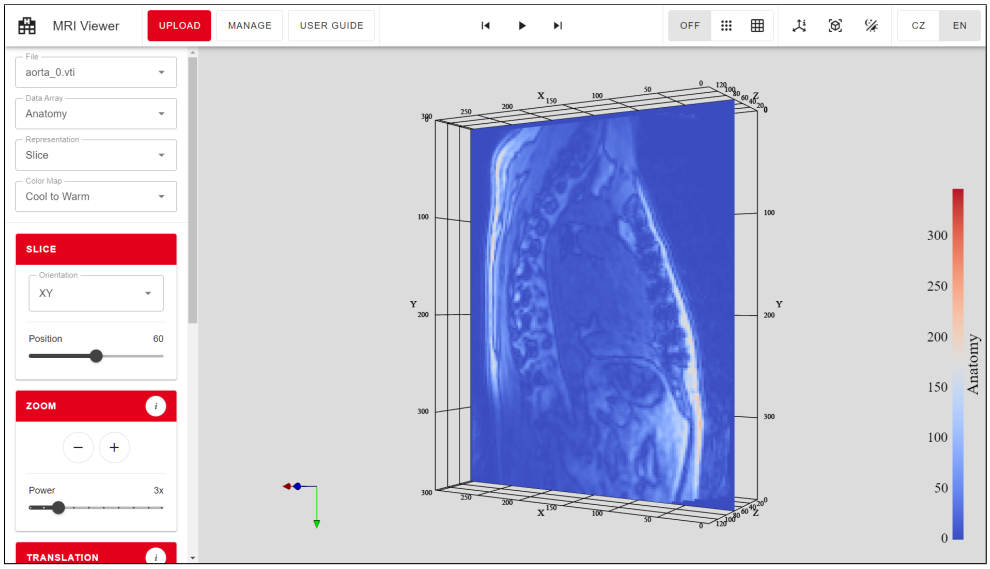

Figure B.9: There is a slice inserted along the X and Y axes and positioned within the data boundaries in the direction of the Z-axis

B.2.7.2 Zoom

This particular tool enables you to zoom the data in or out. The minus icon is for zooming out, and the plus icon is for zooming in. The power determines the depth of zooming. We recommend using the right mouse button for zooming in or out smoothly and quickly rather than using the zoom tool.

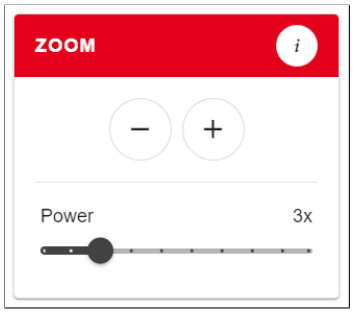

Figure B.10: The zoom tool

B.2.7.3 Translation

This particular tool enables you to pan the data in the direction of any axis. The left arrow icon is for shifting to the negative values of the particular axis. The right arrow icon is for shifting to the positive values of the particular axis. The step determines the length of the shift. We recommend using the middle mouse button for panning smoothly and quickly rather than using the translation tool.

Figure B.11: The translation tool

B.2.7.4 Rotation

This particular tool enables you to rotate the data around any axis. The left-rounded arrow icon is for rotating counterclockwise around a particular axis. The right-rounded arrow icon is for rotating clockwise around a particular axis. You can set the angle of the rotation. We recommend using the left mouse button for rotating smoothly and quickly rather than using the rotation tool.

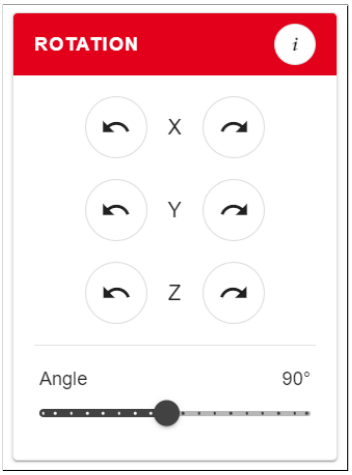

Figure B.12: The rotation tool

B.2.8 Player

A player is available only for groups of files with more than one file. If you choose the file that is included in such a group, then you can start this player by clicking the play icon. The player goes through these files one after another. You can skip to the previous or next file by clicking the side icons (this action also stops the player). You can stop the player by clicking the stop icon that will appear instead of the play icon while playing.

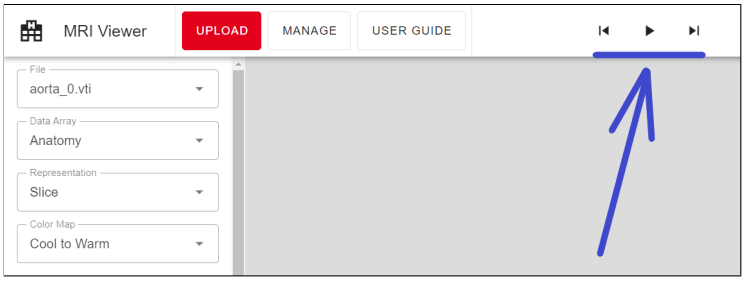

Figure B.13: The player is located in the middle of the upper bar

B.2.8.1 Recommendations

Be patient while playing the first round. The animation might be slow because of the data loading and rendering. The next rounds should be faster.

B.2.9 Point and Cell Information

You can see the values of point or cell data arrays stored in the visualized file by clicking on the particular buttons in the toolbar (*off*, *points*, *cells*).

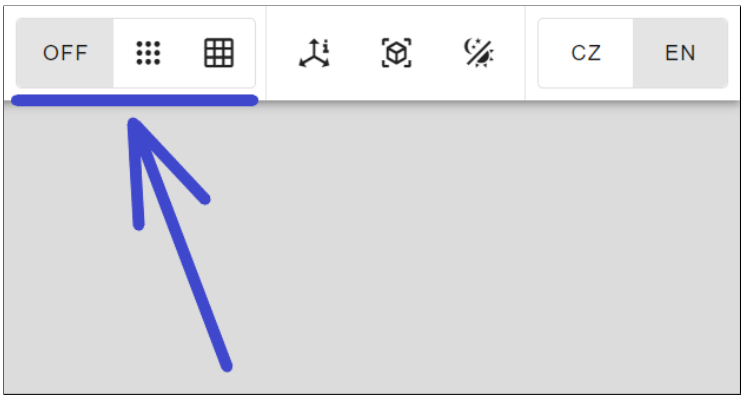

Figure B.14: Point and cell information can be turned on/off in the upper bar

When you click on some point or cell in your data afterwards, the dialog with the values of the particular data arrays will appear. Additionally, there is an identifier and location (for points) and an identifier (for cells). Point or cell information is automatically disabled while playing.

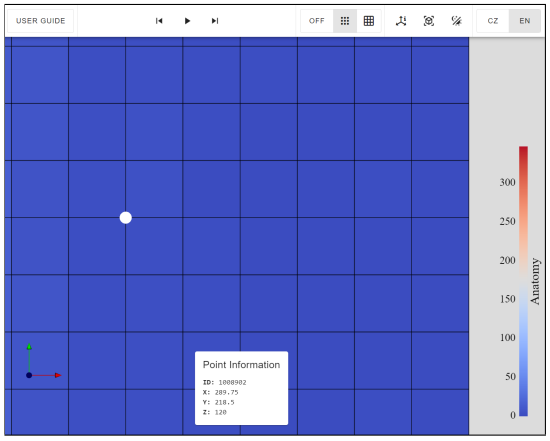

Figure B.15: Point information (i.e., identifier, location, and point data array values, if available) can be found in the bottom part of the screen

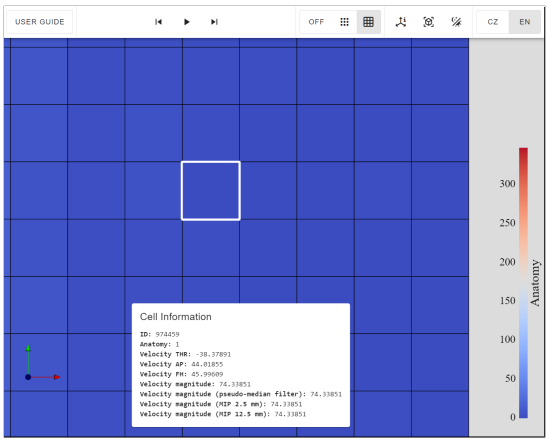

Figure B.16: Cell information (i.e., identifier and cell data array values, if available) can also be found in the bottom part of the screen

B.2.10 Axes Information

You can turn on or off the axes information by clicking on the particular icon in the toolbar.

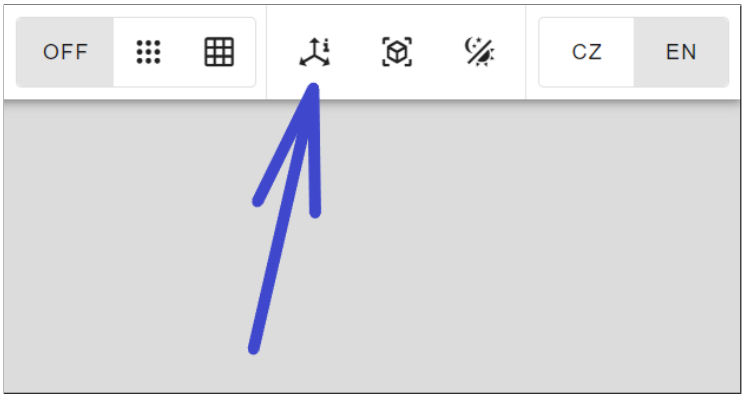

Figure B.17: Axes information can be turned on/off in the upper bar

This will show the axes labels, a grid, and the bounds of your data.

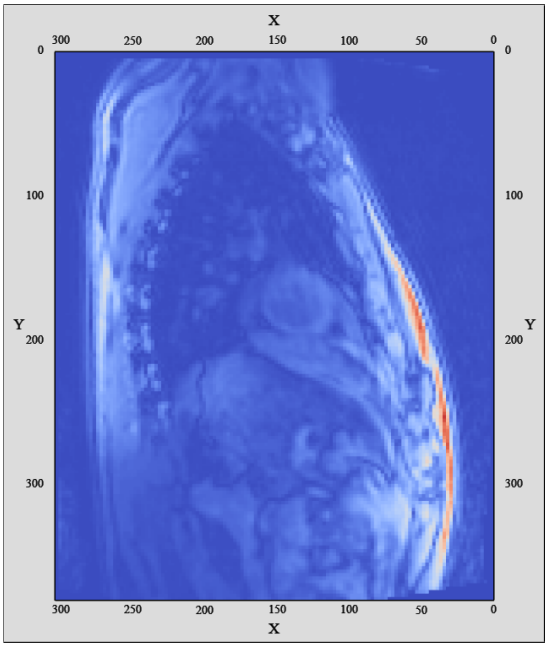

Figure B.18: You have more insight by activating the axes information

B.2.11 Axes Widget

For even better orientation in data and space, there is a small axes widget composed of three basic axes perpendicular to each other. The X-axis is red, the Y-axis is green, and the Z-axis is blue. The axes widget copies the orientation of the data. If you rotate your data, the widget is rotated accordingly.

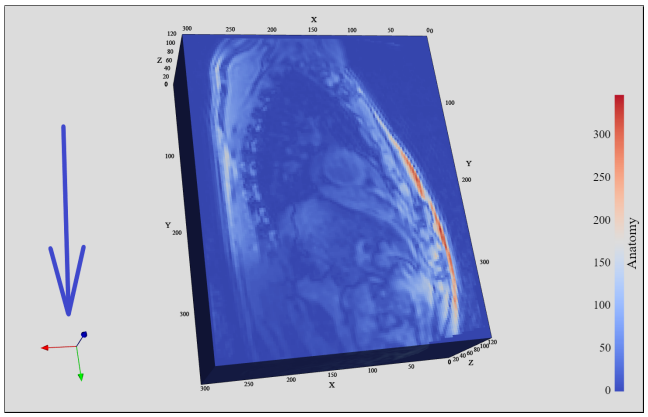

Figure B.19: The axes widget is located in the bottom-left corner

B.2.12 Scalar Bar

There is also another helper on the other side of the screen. The scalar bar maps the data array values to the colors of the selected color map. Therefore, you can determine values in the interesting parts of the visualized data.

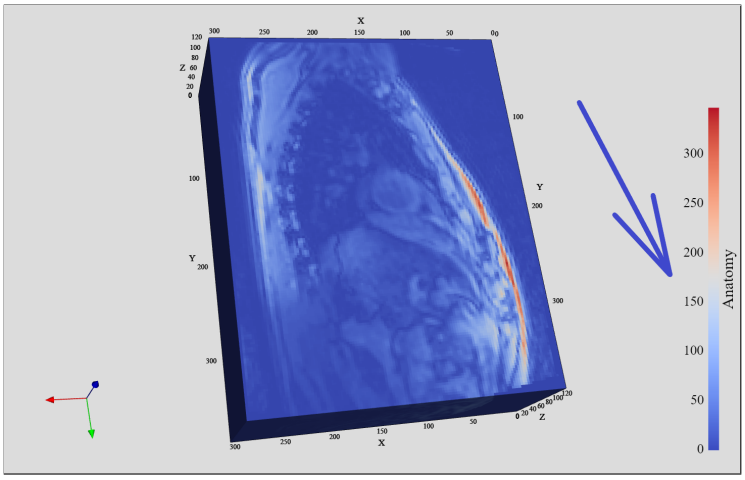

Figure B.20: The scalar bar is on the right side

B.2.13 Reset View

By clicking the *Reset View* icon, you reset all the zoom, translation, and rotation interactions you have made in history.

Figure B.21: The reset view can be found in the upper bar

B.2.14 Dark and Light Themes

You can switch between the light and dark themes by clicking the *Mode* icon.

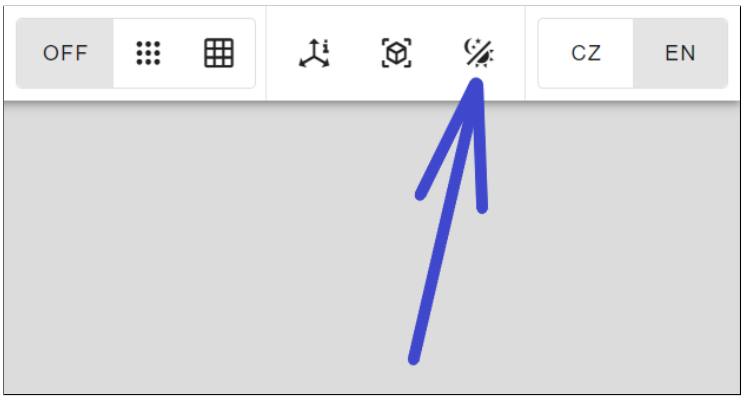

Figure B.22: Mode can be switched in the upper bar

B.2.15 Languages

You can switch between the Czech and English languages by clicking the particular button in the top-right corner.

Figure B.23: Languages can be switched in the upper bar

B.2.16 Progress Bar

When you see a progress bar under the toolbar, then the application is busy working. The progress bar appears while file uploading, data processing, rendering, picking points or cells, switching files, data arrays or representations, and so on. Please be patient and let it disappear. The style of the progress bar indicates the blood flow, as the application is primarily determined for medical purposes.

Figure B.24: The progress bar appears under the upper bar

B.3 Contact

In case of trouble, please contact us at *karelvrabeckv@gmail.com*. Your questions and ideas will make this user guide better for future users. Thank you very much!

Appendix C

Test Scenarios

C.1 Language

The first test scenario is focused on the selection of language. This might be a desired functionality if a user is not familiar with the English language (as the default language of the web application). Therefore, a tester is tasked with selecting the language according to his or her needs.

Expected Time

10–30 seconds.

Covered Features

- Localization.

Preconditions

- The tester has access to the application (either desktop or web). In the case of a desktop application, the *.exe* file (currently around 80 MB) is sent to the tester. In the case of a web application, the URL is communicated to the tester. The choice of the application type depends on the deployment progress in the IKEM.

- The tester has received a package with the test data (i.e., a group of *heart_*.vti* files with blood flow in the human heart and aorta).

Starting Point

The tester launched the application and is waiting for instructions on the startup screen with a dialog for uploading files.

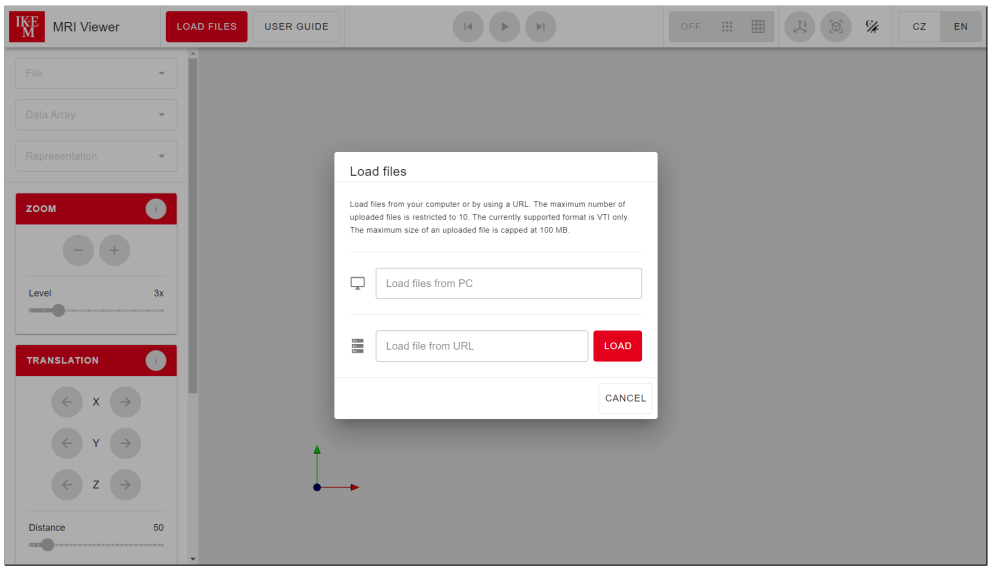

Figure C.1: Starting point in the first test scenario

Instructions

1. Which language is comfortable for you?
2. (if English) Skip to the next test scenario, as English is the default language of the application.
3. (if Czech) Close the dialog for uploading files and switch to the Czech language.
4. (if Czech) Open the dialog for uploading files again.

Expected Steps

1. (if Czech) The tester clicks the *Cancel* button in the dialog.
2. (if Czech) The tester clicks the *CZ* button in the top-right corner.
3. (if Czech) The tester clicks the *Load Files* button in the toolbar.

Ending Point

The same as the starting point.

C.2 Slice Tool

The second test scenario is focused on the slice tool, as it is the most powerful tool in the web application. The slice tool is able to inspect data from the inside. Therefore, the tester is tasked with setting the slice so that the blood flow in the human heart is visible.

Expected Time

1 minute 30 seconds.

Covered Features

- Uploading files from PC;
- Data array selection;
- Representation selection;
- Rotation tool; and
- Slice tool.

Preconditions

- The tester successfully finished the previous test scenario.

Starting Point

The same as the ending point of the previous test scenario.

Instructions

1. Upload *heart_4.vti* from your computer.
2. Set the data array to *Velocity magnitude*.
3. Set the representation to *Slice*.
4. Turn the data upside down.
5. Set the orientation and position of the slice so that the blood flow in the human heart can be seen.

Expected Steps

1. The tester clicks on the *Load files from PC* text field and picks *heart_4.vti* from his or her computer.
2. After the file upload, the tester clicks the *Data Array* select and selects *Velocity magnitude*.
3. The tester clicks the *Representation* select and selects *Slice*.
4. The tester goes to the *Rotation* section in the left column and clicks twice on the rounded right arrow icon to rotate around the Z-axis.
5. The tester leaves the orientation as is and sets the position approximately to the middle of the slider range, which is the spot with a visible blood flow.

Ending Point

The tester can see the blood flow in the human heart and aorta.

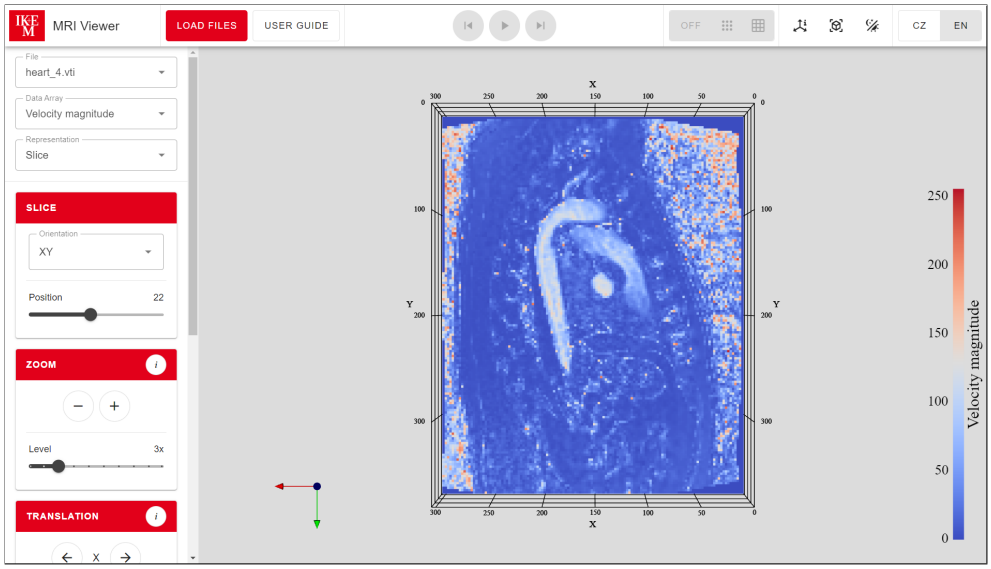

Figure C.2: Ending point in the second test scenario

C.3 Player

The third test scenario is focused on the player, as it is able to play an animation composed of similar *.vti* files. Therefore, the tester is tasked with

uploading the remaining test data and starting the player so that the animation of the blood flow in the human heart can be played.

Expected Time

1 minute 30 seconds.

Covered Features

- Uploading files from PC;
- Player;
- File selection;
- Axes information with grid; and
- Theme switching.

Preconditions

- The tester successfully finished the previous test scenario.

Starting Point

The same as the ending point of the previous test scenario.

Instructions

1. Upload *heart_0.vti*, *heart_1.vti*, *heart_2.vti*, and *heart_3.vti* from your computer.
2. Try switching between the uploaded files.
3. Start the player to play the animation of the uploaded files.
4. Turn off the axes information with grid.
5. Switch the current theme to dark.
6. Stop the player.

Expected Steps

1. The tester clicks on the *Load files from PC* text field and picks the remaining *heart_0.vti*, *heart_1.vti*, *heart_2.vti*, and *heart_3.vti* from his or her computer.

2. After the file upload, the tester clicks the *Previous File* icon or *Next File* icon within the player to switch between the uploaded files. This can also be accomplished by clicking the *File* select in the left column and selecting particular file.

3. The tester clicks the *Play* icon to start the player.

4. The tester clicks the *Axes Info with Grid* icon to turn off the axes information with grid.

5. The tester clicks the *Theme* icon to switch to the dark theme.

6. The tester clicks the *Pause* icon to stop the player.

Ending Point

The tester can replay the blood flow in the human heart.

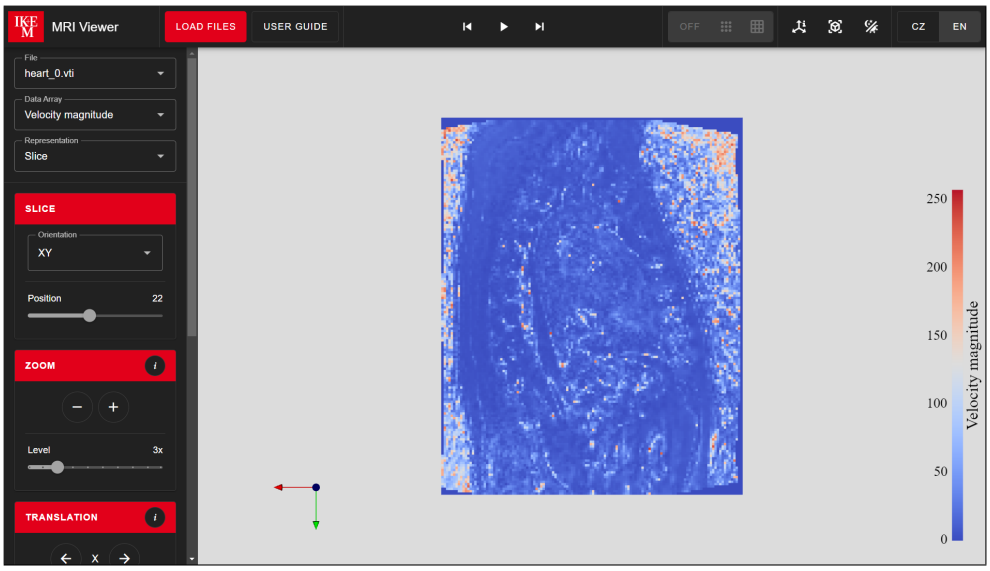

Figure C.3: Ending point in the third test scenario

C.4 Picking

The fourth test scenario is focused on picking points and cells in the interesting parts of the data. Moreover, these data are loaded via URL into the web application. Therefore, the tester is tasked with loading the data via the network and picking some interesting points and cells from them.

Expected Time

2 minutes 30 seconds.

Covered Features

- Uploading files from URL;
- Representation selection;
- Point and cell picking;
- Zoom tool;
- Translation tool;
- Rotation tool; and
- Reset the view.

Preconditions

- The tester successfully finished the previous test scenario.

Starting Point

The same as the ending point of the previous test scenario.

Instructions

1. Load the VTI file from the following URL: `https://data.kitware.com/api/v1/file/5aaf9e688d777f068578dbca/download`
2. With the help of zoom, translation, and rotation tools, focus on the human ear.
3. Set representation to *Surface with Edges*.
4. Turn on the point-picking and pick a random point in the data.
5. Switch to cell-picking and pick a random cell in the data.
6. Turn off picking.
7. Set representation to *Surface*.
8. Reset the view.

Expected Steps

1. The tester clicks on the *Load Files* button, which pops up the dialog for uploading files. After that, he or she inserts the URL into the *Load file from URL* text field and clicks on the *Load* button.
2. After the file fetch, the tester uses the combination of the zoom, translation, or rotation tools from the left column to focus on the ear.
3. The tester clicks the *Representation* select and selects *Surface with Edges*.
4. The tester clicks the *Point Picking* button to turn on the point-picking. After that, he or she selects a random point in the data.
5. The tester clicks the *Cell Picking* button to turn on the cell-picking. After that, he or she selects a random cell in the data.
6. The tester clicks the *Off* button to turn off the picking.
7. The tester clicks the *Representation* select and selects *Surface*.
8. The tester clicks the *Reset View* icon to reset the view.

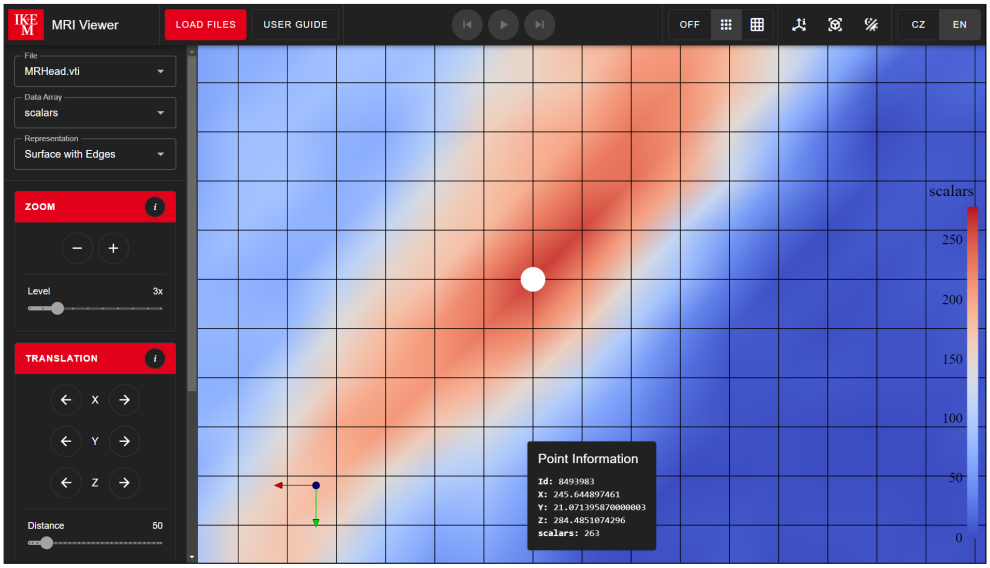

Figure C.4: Picking a random point in the data

Ending Point

The tester can pick points and cells in the interesting parts of the data.

Appendix D

Contents of CD

```
mri-viewer ..... source code
├── wireframes.bmpr ..... wireframes openable in Balsamiq
├── prototype.rp ..... prototype openable in Axure RP
├── thesis.pdf ..... thesis in PDF format
├── thesis.tex ..... thesis in LaTeX format
├── user-guide-en.pdf ..... user guide for international users
└── user-guide-cz.pdf ..... user guide for Czech users
```