

h e g

Haute école de gestion  
Genève

## Creation of a web application using FSL tools



**Bachelor Project submitted for the degree of  
Bachelor of Science HES in Business Information Technology**

by

**Jennifer SCHLAPPINGER**

Bachelor Project Mentor

**Dr. Stefan Kambiz BEHFAR, Professor HEG**

**Genève, 8<sup>th</sup> of May 2023**

**Haute École de Gestion de Genève (HEG-GE)**

**Business Information Technology**

## Disclaimer

This report is submitted as part of the final examination requirements of the Haute école de gestion de Genève (HEG) for the Bachelor of Science HES-SO in Business Information Technology.

The student sent this document by email to the address given by his or her bachelor's thesis mentor for analysis by the URKUND plagiarism detection software, following the procedure detailed at the following URL: <https://www.arkund.com>.

The use of any conclusions or recommendations made in or based upon this report, with no prejudice to their value, engages the responsibility neither of the author, nor the author's mentor, nor the jury members nor the HEG any of its employees.

« I certify that I have carried out the present work alone without the use of sources other than those cited in the bibliography. »

Annemasse, 8<sup>th</sup> of May 2023

Jennifer Schlappinger

## **Acknowledgements**

First of all, I would like to express my deepest gratitude to my professor Stefan Behfar for his patience, availability and support. This endeavour would not have been possible without him.

A special thank you go to my mother, my sister and my father, who supported me emotionally and practically throughout this entire writing process. Last but not least, I'd like to thank the Lord Jesus for giving me strength all along this journey.

# Abstract

Nowadays, non-invasive scanning procedures, such as MRI, allow scientists to look at the brain in more detail and thereby detect connections between various brain regions that - when put together - construct an entire brain network. They do not do that by themselves but with the help of medical software applications, which will briefly be reviewed in this thesis for their functionalities and their technical characteristics. Using the well-documented FSL application, the goal of this thesis is to conceptualize and finally implement a medical web application that creates a brain connectivity matrix that is user-friendly and easily extensible, creating an application framework for future, more automated research about Alzheimer's disease.

# Table of contents

<b>Disclaimer</b> .....	<b>i</b>
<b>Acknowledgements</b> .....	<b>ii</b>
<b>List of tables</b> .....	<b>vi</b>
<b>List of figures</b> .....	<b>vi</b>
<b>List of codes</b> .....	<b>vii</b>
<b>List of abbreviations</b> .....	<b>vii</b>
<b>1. Introduction</b> .....	<b>1</b>
<b>1.1 Thesis objective</b> .....	<b>2</b>
<b>1.2 Thesis structure</b> .....	<b>2</b>
<b>2. Background</b> .....	<b>3</b>
<b>2.1 MRI Images</b> .....	<b>3</b>
2.1.1 The difference between fMRI and sMRI .....	4
<b>2.2 Preprocessing</b> .....	<b>5</b>
<b>2.3 Analysis</b> .....	<b>5</b>
2.3.1 Network analysis.....	6
2.3.2 Building a brain connectivity matrix (BCM) .....	7
<b>2.4 MRI Application Software</b> .....	<b>7</b>
2.4.1 Web application .....	8
2.4.2 MRI analysis software .....	10
2.4.3 FMRIB Software Library .....	11
<b>3. Research model</b> .....	<b>13</b>
<b>3.1 FSL-Web-Application</b> .....	<b>13</b>
3.1.1 Functionalities .....	13
3.1.2 Analysis of the advantages linked to a web application .....	16
3.1.3 Future vision .....	17
<b>4. Implementation</b> .....	<b>18</b>
<b>4.1 Technical specifications</b> .....	<b>18</b>
<b>4.2 Technical implementation choices</b> .....	<b>19</b>
4.2.1 Frontend – Javascript with React .....	19
4.2.2 Backend – Python with Flask, Nypipe and Oct2Py .....	20
4.2.3 Database - PostgreSQL .....	21
<b>4.3 Software design</b> .....	<b>22</b>

4.3.1	Analysis .....	25
4.3.2	Preprocessing .....	31
4.3.3	User management.....	35
<b>4.4</b>	<b>Tests and output .....</b>	<b>41</b>
4.4.1	Testing requirements .....	41
4.4.2	Outputs and results .....	43
4.4.3	Test data.....	49
<b>4.5</b>	<b>Installation and employment.....</b>	<b>50</b>
4.5.1	Installation.....	50
4.5.2	Employment.....	50
<b>5.</b>	<b>Conclusion .....</b>	<b>51</b>
	<b>Bibliography .....</b>	<b>52</b>
	<b>Appendix 1: Installation manual for developers.....</b>	<b>58</b>

## List of tables

Table 1: Preprocessing steps [13] .....	5
Table 2: Comparison desktop application vs web applications .....	9
Table 3: Number of references obtained from Google scholar website until 05-May-2023 and other software specificities [5, 33, 34, 35, 36] .....	10
Table 4: Steps to generate a BMC .....	12
Table 5: Functionalities .....	14
Table 6: Basic functionality of the web application.....	15
Table 7: Coding conventions frontend.....	20
Table 8: Coding conventions backend.....	21
Table 9: Backend - coding conventions.....	21

## List of figures

Figure 1: MRI image of a human brain [14] .....	3
Figure 2: Acquisition process of fMRI [16] .....	4
Figure 3: Time series extraction [17] .....	4
Figure 4: Connectivity matrix [13 – Main Course Material, Ch. 34]] .....	6
Figure 5: Construction of a BCM [20] .....	7
Figure 6: Web applications basic architecture [51] .....	8
Figure 7: Global Internet Users Over Time January 2022 DataReportal [29] .....	8
Figure 8: Drawn illustration of an EEG [50].....	17
Figure 9: Future functionalities .....	22
Figure 10 : Backend design.....	23
Figure 11: Database model .....	24
Figure 12: User interface of FSL Nets .....	25
Figure 13 : All processes linked to one user .....	26
Figure 14 : Result folder of a network analysis .....	27
Figure 15 : Automated BCM output .....	27
Figure 16: Backend creation of a BMC sequence diagram .....	28
Figure 17: GUI Preprocessing .....	31
Figure 18: Sequence diagram preprocessing .....	33
Figure 19: GUI Login .....	35
Figure 20: Backend - Verification of credentials [67].....	36
Figure 21: Backend - Verification of the session token [67] .....	36
Figure 22: Frontend Registration .....	37
Figure 23: Registration service .....	38
Figure 24: Frontend, forgot password.....	39
Figure 25: Password reset request.....	40
Figure 26: Successful backend pytest test .....	42
Figure 27: Failed backend pytest test.....	43
Figure 28 : BCM parameters .....	43
Figure 29 : BCM A, changing parameters 1 and 2.....	45
Figure 30 : BCM B and C, changing parameters 1 and 2 .....	45
Figure 31 : Process' names in database.....	45
Figure 32 : Invalid process name (not unique).....	45
Figure 33 : BCM with 5 nodes (right) and 44 nodes (left).....	46
Figure 34 : BCM - full correlation .....	46
Figure 35 : BCM - Partial correlation .....	47
Figure 36 : BCM - full correlation, with r to z .....	47
Figure 37 : BCM - Covariance.....	47
Figure 38 : BCM - Partial ridge regression.....	48
Figure 39 : Preprocessing - brain extraction .....	48
Figure 40 : BET extracted brain.....	49

Figure 41 : Mail when requesting a password reset.....	49
--	----

## List of codes

Code 1: Creation of tables.....	24
Code 2: Frontend - Get TR.....	26
Code 3: Backend - Octave interface.....	29
Code 4: octave script, named createMatrix.m.....	29
Code 5: Frontend - Initial form.....	31
Code 6: Frontend - Parameters.....	32
Code 7: Frontend - Parameter submission.....	32
Code 8: Backend - Preprocessing.....	34
Code 9: Backend - Password encryption.....	37
Code 10: Backend - Recieving encoded password from database.....	37
Code 11: Database - Hashed password.....	37
Code 12: Retrieving the user's mail from database with python.....	39
Code 13: Set up of a correct test.....	42
Code 14: Set up of an incorrect test.....	42

## List of abbreviations

API	Application Programming Interfaces
BCM	Brain Connectivity Matrix
CL	Command Line
DICOM	Digital Imaging and Communications in Medicine
fMRI	functional MRI
FMRIB	Functional Magnetic Resonance Imaging of the Brain
FSL	FMRIB Software Library
GUI	Graphical User Interface
MRI	Magnetic Resonance Imaging
OS	Operating System
sMRI	structural MRI (sometimes also aMRI, for anatomical MRI)
TR	Time series of an fMRI



# 1. Introduction

Alzheimer's disease is a well-known neurodegenerative disorder leading progressively to memory deficits and thereby causing dementia - it mainly affects elderly persons [1]. Until this day, no definitive cure has been found, but treatments are available [2,3]. Since Alzheimer's disease causes the death of the neuron's connections [3], research concerning Alzheimer evolves primarily around brain analyses. For Alzheimer, in particular, the network analysis is extremely helpful since it correlates different brain regions and indicates each connection's intensity. The connectivity matrix resulting from the network analysis, pinpoints alterations of the neurological network compared to a healthy brain's network and thus allows quicker identification of the disease in its earlier stages [1,4].

In spite of the easily understandable theoretical idea of such network analyses, practically generating a brain connectivity matrix (BMC) with some Magnetic Resonance Imaging (MRI) images only, can be complex and requires prior knowledge of the subject, which includes the building and acquisition of an MRI image, the preparation and processing methods of MRI images prior to analysis and finally how these images can be inspected and interpreted correctly. In practice, these fundamental concepts are always realised with the help of a technical device such as an MRI-Scanner, MRI image software and other appliances. For network analyses, in particular, functional MRI (fMRI) -image applications, which enable the user to apply single-stepped procedures on the acquired data depending on the desired output, are typically used or even required [5]. The medical software available around fMRI- and MRI images of the brain are diverse [5,6]. Hence, the selection of fitting software depends on various factors, such as the medical application field, the type of processes to be executed, the technical requirements of the software itself, the costs linked to the software acquisition and maintenance, and ultimately even personal preference can play a role [6].

The adequate application choice is also relevant when daring a glance at what lies ahead in a time when artificial intelligence (AI) could significantly contribute to the early detection of Alzheimer's disease [3]. AI commonly deals with massive data flux containing diverse data formats. This phenomenon, across many other domains as well, is referred to as big data, which has characteristically data streams that are high in volume, velocity and variety (3 Vs) [7]. Why is this important? Because it highlights the need for an application that has a powerful performance - or at the very least, can be adapted conformingly in the foreseeable future. A tool that easily manages the process of generating network connectivity matrices and simultaneously uses AI to handle a large input of data, could eventually find patterns of brain alterations in the matrices and predict

signs of Alzheimer in a more automated way opening the door for even more research in that area.

Simultaneously, these applications should meet the continuously higher standard of the user's requirements. A user-friendly interface and online-accessibility has become a must for new applications [8].

However, such a tool does not exist yet. Consequently, in the scope of this thesis, an answer to the following question is sought: Can a basic, but extensible framework for a web-application that generates network connectivity matrices, to allow future research concerning Alzheimer's disease based on the methodology of brain networks be built?

## **1.1 Thesis objective**

The project aims to develop a web application

- that allows a user from a non-medical background to without any further need for profound instructions
- to view and have simple analyses performed on MRI-brain-images
- which takes MRI images as the principal input element and
- generates brain connectivity matrices as the principal output element

## **1.2 Thesis structure**

To do so, the following chapters will explore the fundamental concepts of MRI imaging to understand how a network connectivity matrix is built in theory. In the next step, we will look at the applications of the medical field in general from a technical point of view to better understand the different software for MRI preprocessing and analysis available. The FMRIB Software Library (FSL) library will be the software application with which this thesis will work. Once the brain connectivity matrix and MRI imaging concepts are understood, they will be brought together and with the help of FSL, a step-by-step reference pipeline will be set up. It is then, that the utility of an FSL web application will be analysed. Eventually, the conceptualisation of the subsequent implementation will be described, and the thesis will be concluded.

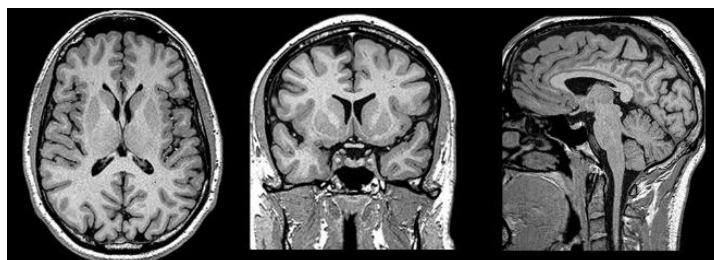
## 2. Background

Analysing the brain is nearly as complex as the brain itself. This is why this chapter is entirely dedicated to delving into the basic concepts of MRI images and their application methods. Especially in earlier days, the person's death was inevitable if his/her brain should be analysed [9]. Thankfully today, various non-invasive methods have been discovered, such as positron emission tomography (PET), near-infrared spectroscopy (NIRS), magnetoencephalogram (MEG), electroencephalography (EEG) and functional and structural magnetic resonance imaging (sMRI), to only name a few [10]. The latter, one of the most recently developed forms of neuroimaging, is the technique this thesis will focus on.

### 2.1 MRI Images

As evoked above, MRI stands for magnetic resonance imaging and allows the brain to be analysed in a non-invasive way. This is achieved due to the energy released by our bodies' protons when they interact with the magnetic field and radiofrequency pulses in the MRI machine. This generated energy is registered by the connected computer, which then calculates a three-dimensional image composed of various voxels [11]. Voxels are the entity of a cubic volume of a 3-dimensional computer-generated space, similar to pixels in a 2-dimensional space [12]. The voxels of an MRI image could be either whitish if that area had a high signal intensity, hyperintense, or darkish if that area had a low signal intensity, hypointense. The "white matter" connects the neurons to each other and conducts impulses away from the soma. They are the output entity of electrical signals sent within the central nervous system and are commonly an indicator of a neuro-chronic disease. The "grey matter," on the other hand, is mainly made up of neuron cell bodies, neuron somas. They are the input unit of electrical signals sent within the central nervous system. Lastly, when examining an MRI image, hollow spaces are noticeable. These are spaces filled with Cerebrospinal fluid (CSF) and are usually referred to as the "third tissue" [13].

Figure 1: MRI image of a human brain [14]



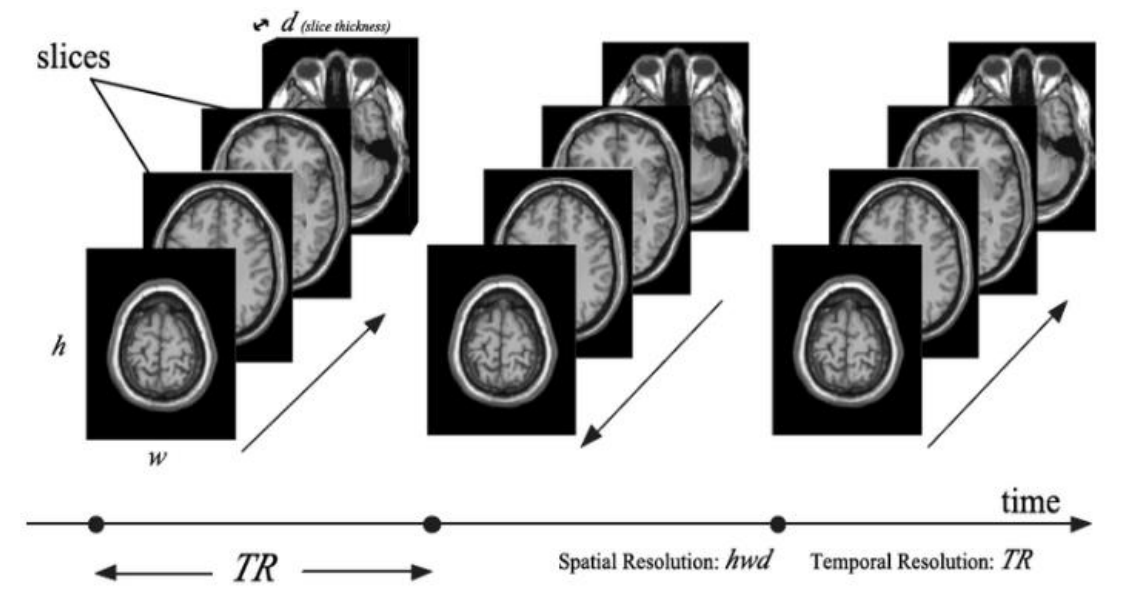
Each image, also brain volume, consists of roughly 100'000 equally sized voxels or volume elements that are spatially located [12].

### 2.1.1 The difference between fMRI and sMRI

Structural magnetic resonance imaging (MRI) examines the anatomy and pathology of the brain - as opposed to functional magnetic resonance imaging [fMRI], which examines the brain activity while the subject is performing a specific task – actively or passively. Furthermore, structural MRIs provide an anatomical reference for visualizing the activation patterns and regions of interest (ROI) to extract helpful signal information [13]. In fMRI, the brain areas which were active during the performed task are highlighted due to changed blood oxygenation and blood flow. It is differentiated between resting state fMRI and task fMRI, even if they are often used complementary [15].

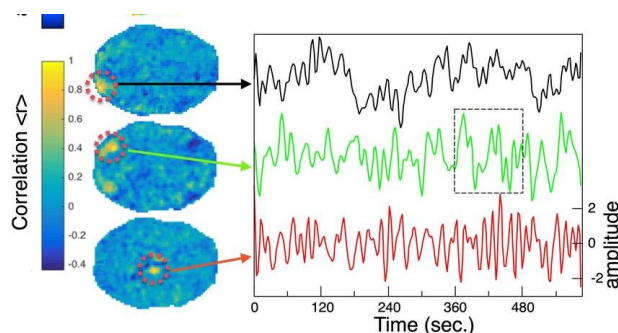
When registering an fMRI multiple brain volumes are repeatedly “photographed” about every two seconds [12]. This time lapse from one registration to the next is called **TR = Repetition Time**, while the concatenated string of all volumes is named a run of data [16].

Figure 2: Acquisition process of fMRI [16]



Another essential concept is the time series extraction which englobes the process of zooming in on a single voxel and lining up its intensity according to each volume across time, as seen in the next Figure 3.

Figure 3: Time series extraction [17]



Finally, when putting these notions into correlation, one slice of Figure 2 would figuratively correspond to one of the regions marked with a red-dotted circle in Figure 4. And each line of the table in Figure 3 shows the variations of the voxel's intensity throughout the fMRI session.

## 2.2 Preprocessing

The freshly generated MRI-Image cannot be analysed immediately but must first undergo a "cleaning process" called preprocessing. Several factors can distort the outputs of an MRI scanning session and thus falsify the results. They are referred to as noise and can have multiple sources, such as [18, 13]:

- Minor movements of the person influencing the final MRI
- Heartbeat, respiration, low-frequency oscillations
- The complexity of the brain structure, tissues are not clearly distinguishable
- Technical issues in the scanner, so-called scanner artefacts

Typically, experienced professionals will apply a step-by-step procedure (also: workflow or pipeline) best adapted to target and eliminate the noise by removing unwanted movements of the image, adding depth and detail, and smoothing other technical disturbances. Moreover, re-arranging each step's order may influence the final image's outcome.

Table 1: Preprocessing steps [13]

Conventional preprocessing steps	Special preprocessing steps
Motion and distortion correction	Nuisance regression
High pass temporal filtering	Global signal regression
Slice time correction	Volume censoring
Spatial smoothing	ICA-based clean-up
	Low pass temporal filtering

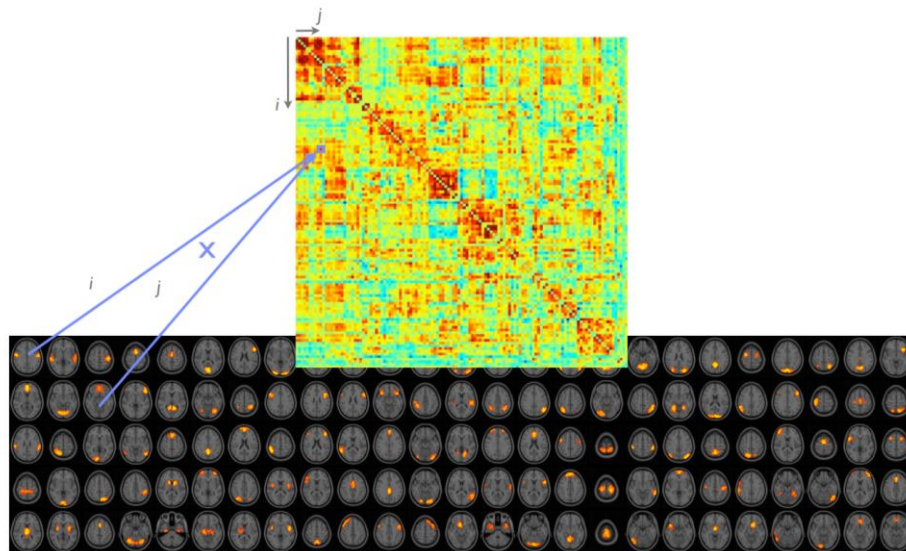
## 2.3 Analysis

Once the preprocessing is done, the images can be analysed depending on the MRI-type. As mentioned in the introduction, the web application should create a brain connectivity matrix. A network analysis must be performed on the desired data to do so. This type of analysis is often performed on resting-state fMRI and describes brain functions by how strongly the individual brain regions are interconnected [19]. These connections are contextualized and visualized by creating a brain connectivity matrix for deeper understanding.

### 2.3.1 Network analysis

As with any other type of networks, the following two fundamental elements must be given, for a network to be classified as such: the nodes and the edges. That is not different for the brain connectivity matrix. While the node is represented by a region or a parcel of the brain, sometimes also called ROI, the edges will show the strength of the interrelation between these processing units. Each column and row corresponds to one predefined brain region [13]. For Figure 4, the stronger the connectivity between two nodes, the darker the intersections in the matrix. Directionality will not be considered in the scope of this project. The diagonal top-left to bottom-right is greyed out, as it points to the same node on both axes.

Figure 4: Connectivity matrix [13 – Main Course Material, Ch. 34]



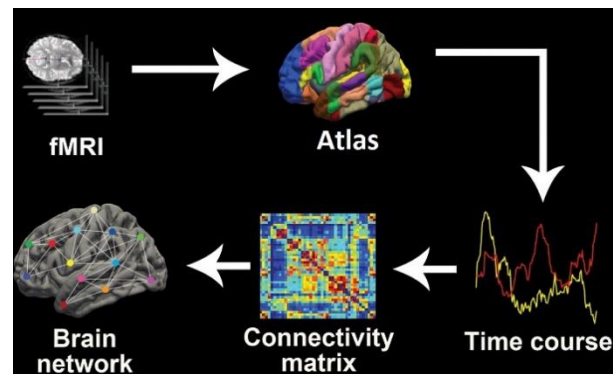
#### 2.3.1.1 Brain parcellation

Brain parcellation, or node definition, is the process that will split the brain into multiple ROIs. These regions can either be contiguous or non-contiguous. If contiguous, each node will be a "real" area of the brain, whereas a non-contiguous node will consider a node as one of multiple parts of the brain that are logically linked. Moreover, nodes can be defined as either binary or non-binary. With a binary approach, the brain undergoes a hard parcellation, meaning every voxel is clearly attributed to one voxel. In a non-binary approach, the brain has a weighted parcellation; thus, not every region, especially a node's borders, is equally counted. The notion of atlases serves as a roadmap/reference map for the detected ROIs. They are usually established by using sMRIs. Applications of these atlases in the process of a network analyses must be considered thoroughly. [13].

### 2.3.2 Building a brain connectivity matrix (BCM)

How is a connectivity matrix built? First, as seen in Figure 5 when the fMRI timeseries is completed, the brain will be parcelled using a desired atlas.

Figure 5: Construction of a BCM [20]



Once parcellation completed, the time series of each region will be extracted and averaged. The extracted time course looks similar to the one shown Figure 7, at step 3. The edges are then calculated by putting every node into correlation with each other. The time series is always linked to one node [20].

In summary, to create a BMC the following steps must be performed:

(0) Data acquisition → (1) Preprocessing → (2) Node definition → (3) Timeseries extraction → (4) Timeseries averaging → (5) Edge calculation → (6) Creation of network matrix [13, 20, 21]

## 2.4 MRI Application Software

Brain analyses are complicated procedures, mainly when performed on living subjects. Several appliances, applications and machines have been developed by now that facilitate these complex processes for medical professionals. Medical applications are the type of support this thesis will focus on. According to Law insider a medical application is defined as follows:

*Medical Applications means diagnostic products, therapeutic and prophylactic drugs or vaccines, intended for the diagnosis, prevention, or treatment of disease in humans, animals or plants and all discovery, research, development and commercialization efforts to support those uses, including without limitation, elucidation of gene function and target validation. [22]*

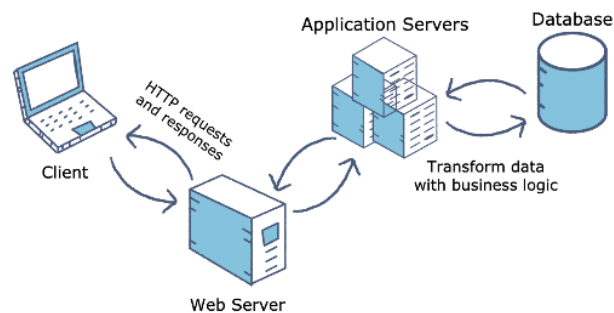
As the definition above implies, medical applications can be very vast, and finding an application that suits all the user's needs and requirements can be difficult.

From a technical point of view, however, we can differentiate between different types of applications, as we will see in the following chapter.

## 2.4.1 Web application

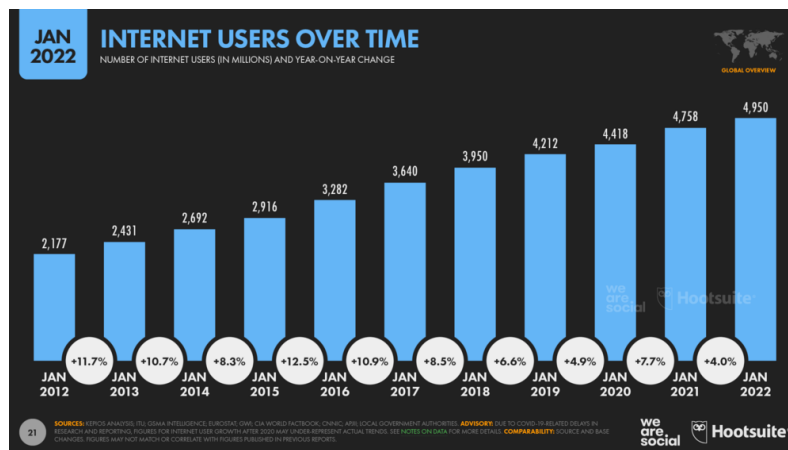
A web application is a computer program that utilizes web browsers and web technology to perform tasks over the internet. A web application is accessible via a web browser. It is, hence, irrelevant on what operating system (OS) the computer is running. The user simply needs an internet connection [26]. If the web application is accessed via its Uniform Resource Locator (URL), the web server will execute a script that is typically based on HTML and JavaScript [27]. If the website is static, no additional process is required. Nowadays, most applications are dynamic, meaning they will need to send specific requests to an application server, which will process the request by also interacting with other services and databases. Some popular web applications are Google Apps that include Gmail, Google Docs, online storage and more [28].

Figure 6: Web applications basic architecture [51]



There are no content-based limitations of web applications. Web applications are getting more popular day by day. This does not come as a surprise. According to the Digital 2022 Global Overview Report, the number of active internet users has gradually increased over the last ten years [29]. The use of web application, and thus the demand for web applications will therefore also grow conformably.

Figure 7: Global Internet Users Over Time January 2022 DataReportal [29]





The major counterpart to web applications are the native applications such as the FSL application, though no clear line can be drawn between all different types of application. In the following Table 2 a quick comparison between web and native applications is made.

Table 2: Comparison desktop application vs web applications

	Web application	Native application
<b>Installation</b>	No installation needed.	Before being able to use the desktop application, it must be installed locally on the computer.
<b>Updates</b>	Updates are done on the server side only. The user does not have to do anything.	Each update must be done on the specific machine, on which the application was installed.
<b>Access</b>	The application can be accessed via a web browser. An active internet connection is a prerequisite.	The application can only be accessed on the machine the applications was previously installed.
<b>OS</b>	The operating system is irrelevant for the web application. It runs on the web server.	The desktop application will run on the operating systems it was coded for. (Example: FSL can only be run on Linux based OS).
<b>System requirements</b>	All the application processes and calculations are run on the application or web server. Therefore, only the latter two must be up to date.	The machine on which the application is desired to be used, must be enough performant to comply with the applications demand, since it is the machine itself that will process the applications calculations.
<b>Security</b>	Considering the fact that the application is running via the internet, everyone can access it – also hackers. Web applications are consequently more vulnerable to a possible hacker attack.	Desktop applications are not likely to be hacked, since the hacker must have direct access to the machine.
<b>Performance</b>	If the internet connection is weak or instable, the application's performance can be slowed down. Moreover, depending on the way the web application is built, the RAM memory, the type of processor and the amount of your	The application can directly interact with the machine and therefore use its resources in the most optimized way possible.

	cache memory may impact the application's performance.	
--	--	--

[30, 52]

Finally, hybrid applications unite features of both worlds. While running on a web browser (the latter is invisible to the user) they can access native platform on the device itself. They are typically installed on the device, similarly to native applications [31, 32].

## 2.4.2 MRI analysis software

In a 2012 article from Mehdi Behroozi and Mohammed Reza Daliri reviewed the different medical software tools that allow preprocessing and analysing of MRI data, which gives an overview on the vast application landscape [5]. In the scope of this thesis only the four most conventional applications [5, 6] will be considered further on.

Table 3: Number of references obtained from Google scholar website until 05-May-2023 and other software specificities [5, 33, 34, 35, 36]

Software Name (Reference to the original paper [5])	Nb. of references		Source code availability	OS	Type of app
	08.2012	05.2023			
SPM: Statistical parametric maps in functional imaging: A general linear approach. [72]	6095	11069 (+81.6%)	Open source	Linux, Mac OS and Windows*	Native, written in MATLAB
AFNI: Software for analysis and Visualization of Functional Magnetic Resonance Neuroimages. [73]	3055	10764 (+252.3%)	Open source	Linux an Mac OS, **	Native, written in ANSI C
FSL: Advance in functional and structural MR image analysis and implementation as FSL. [74]	1909	13238 (+593.5%)	Open source	Linux and Mac OS, **	Native, written in C++
BrainVoyager: Analysis of FIAC data with BrainVoyager QX: From single-subject to cortically aligned group GLM analysis and self-organizing group ICA. [75]	295	1062 (+260.0%)	Closed source	Linux, Mac OS and Windows	Native, written in C++

\*runs on Windows only when installing additional features, \*\*runs on Windows only over a virtual environment

Each of one of the applications cited in Table 3 could witness an increasing number of references, according to Google Scholar over the course of the last ten years, notably FSL, which overtook the in 2012 most cited SPM application. Except for BrainVoyager all the other applications are open source, which means that the source code is available and accessible for free [37]. On the other hand, BrainVoyager is the only tool that can efficiently, be run on Windows, which is the dominant OS worldwide [38]. Consequently, to use FSL, AFNI or SPM, a Windows user will either need to run the application on a

virtual Linux machine or find other solutions like investing in a new Linux computer, for example. All these applications are native applications.

In regard to the software features, all four applications support preprocessing and statistical analysis and output. They can process 2-dimensional, 3-dimensional and 4-dimensional data input, display it and allow the user to manipulate the image view. While all five applications support the output of brain regions, only Brain Voyager and FSL include features for ROI analyses [5, 35]. It can be concluded that FSL is the only tool, that is open-source and that comes with features for ROI analyses, which is especially important in the context of creating a network matrix. Moreover, having experienced the most significant growth of reference, utilisation guides and supporting documentation is amply available.

### 2.4.3 FMRIB Software Library

According to the official FSL website, FSL is described as below:

*“FMRIB Software Library (FSL) is a comprehensive library of analysis tools for FMRI, MRI and DTI brain imaging data. Most of the tools can be run both from the command line and as GUIs (“point-and-click” graphical user interfaces).” [35]*

The scripts from the FSL library can be run on either Linux or macOS. Its first stable release was in March 2019 and most of its code is open source.

FSL is an extremely powerful tool when it comes to applying and automating workflows. It does so, by unifying some of the most crucial preprocessing and analysis steps into one single pipeline. The person handling the data must not run the steps manually and thereby the entire workflow is simplified.

The software library is split as follows [42]:

Functional MRI;	FEAT, MELDOIC, FABBER, BASIL VERBENA, FSLNets
Structural MRI;	BET, FAST, FIRST, FLIRT & FNIRT, FSLVBM, SIENA & SIENAX, MIST, BIANCA, MSM, fsl_anat
Diffusion MRI;	FFT, TBSS, XTRACT, edgy, topup, eddyqc
GLM / Statistics;	FSLeyes, FSLView, Fslutils, Atlases, Atlasquery, SUSAN, FUGUE, MCFLIRT, Miscvis, POSSUM, BayCEST, ICA_PNM, FSL-MRS

It also comes with a handful of data, that are typically used as masks or reference-MRI-files during the different steps of the workflow.

As mentioned, multiple ways exist to set up a network analysis pipeline. The following Table 4 will serve as a reference of the minimum steps recommended for how a BMC can be generated.

Table 4: Steps to generate a BMC

Step	Process	FSL-Tool	Utility
0	Data acquisition		<i>Not covered in this thesis.</i>
1	Preprocessing	FEAT, BET, MCFLIRT	Typical preprocessing steps for a network analysis are Slice time correction (FEAT), motion correction (MCFLIRT) and brain extraction (BET). However, depending on the analysis, other preprocessing steps might be useful.
2	Node definition	Melodic	Responsible for the brain's parcellation. This can be done with the help of an atlas, that in the context of FSL is also named brain mask.
3	Time series extraction	dual_regression	The time series will be extracted from each node, that has been defined in the previous step.
4	Timeseries averaging	FSLNets	Optionally, the extracted and averaged timeseries can be cleaned to avoid unrealistic nodes.
5	Edge calculation	FSLNets	Calculates the strength of the correlations from each node $n_{[0]} \rightarrow n_{[1-n]}$
6	BCM generation	FSLNets	Visual construction of the BCM.

[13]

This pipeline will serve as the framework for the rest of this thesis. It is important to underline that each step could be realised in a different way and order, according to the user's need and still generate a BMC in the end.

In FSL these pipelines can be automated by writing personal scripts that are read and interpreted by the FSL application.

As for now, there is no medical web application yet that runs preprocessing and brain analyses.

### 3. Research model

The FSL tools are versatile and with the help of other software such as Octave and MATLAB, brain connectivity matrices can be generated. It does require though, profound background knowledge of each of these tools plus a lot of time and effort to set these applications up. These are all considerable burdens that make it almost impossible for a person without any medical background to create such a matrix that could, however, provide deeper insight into the brain's health state - not only to the patient oneself but possibly also to the medical professionals. Implementing all these functionalities into web application will not only solve these above mentioned issues but also provide additional advantages that will be explored in this chapter.

#### 3.1 FSL-Web-Application

Building a web application comes with a lot of advantages as seen in the previous chapter. Yet, the FSL-Web-Application will not be a copy-paste of the already existing FSL application, but rather a user-friendly adaptation, which has, in a first time, the priority to easily conduct a network connectivity analysis and the preprocessing steps that are highly recommended when doing such an analysis. Furthermore, it should be possible to extend the application in a potential future adaptation and additions of other analysis processes, such as EEG and NPT (see chapter 3.1.3 Future vision). In order to structure all the needs around the application, a list of the functionalities that are wished to be implemented must be established first.

##### 3.1.1 Functionalities

The FSL library is large. Hence, implementing the entire library into the web application would go way beyond the scope of this thesis. The functionalities will therefore be limited to the FSL tools required to generate the BCM only. The functionalities are listed in the Table 6 below. They are ordered from most to least important and grouped by the functionality category. The application contains four functionality categories which are:

- Analysis: all functionalities related to the generation of a BCM
- Preprocessing: all functionalities related to FSL's preprocessing
- User space: all functionalities related to access personal processed data
- User management: all functionalities related to the user's role and rights

The method used to hierarchise the functionality by its priority is the MoSCoW Prioritization tool. It can be defined as follows:

*“MoSCoW prioritization is a tool for establishing a hierarchy of priorities during a project. It's based on the agile method of project management, which aims to*

*strictly establish factors like the cost of a product, quality and requirements as early as possible. “MoSCoW” is an acronym for must-have, should-have, could-have and won't-have, each denoting a category of prioritization.” [49].*

Table 5: Functionalities

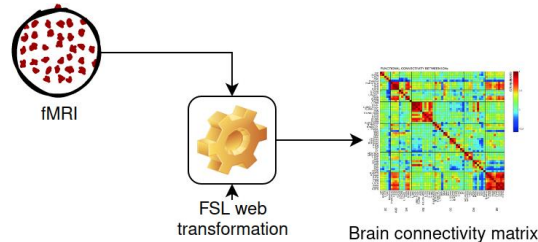
ID	Group	Functionality	Prioritization
1.1	Analysis	Generate simple BCM	Must-Have
4.1	User space	Download generated BCM	Must-Have
2.1	Preprocessing	Slice time correction	Should-Have
2.2	Preprocessing	Motion correction	Should-Have
2.3	Preprocessing	Brain extraction	Should-Have
4.2	User space	Download preprocessed files	Should-Have
3.1	User management	Login	Should-Have
3.2	User management	Logout	Should-Have
3.3	User management	Registration	Should-Have
3.4	User management	Change password	Should-Have
3.5	User management	Admin: Activate and deactivate user	Should-Have
1.2	Analysis	Personalised node definition	Could-Have
1-3	Analysis	Add personalised labels to BCM	Could-Have
2.4	Preprocessing	Spatial smoothing	Could-Have
2.5	Preprocessing	Highpass filtering	Could-Have
4.3	User space	Store and view past analyses results	Could-Have
2.6	Preprocessing	Change order of preprocessing pipeline	Wished-Have
4.4	User space	Save preprocessing and analysis scripts	Wished-Have
4.5	User space	View files	Wished-Have

### 3.1.1.1 Must-Haves

*“The Must-Haves are the essential features that need to be included in the product. Failing to include one would result in a failed release” [49]*

In its most basic usage, the application must be able to process one fMRI file as an input into a brain connectivity matrix as an output (single subject analysis).

Table 6: Basic functionality of the web application



Thereby, it can ignore any additional or optional parameters and preprocessing steps.

### 3.1.1.2 Should-Haves

*“Should-Haves are important requirements but not essential. They are initiatives that are of great importance and add significant value, but are not crucial” [49]*

In a more developed version of the application, the user can decide which of the preprocessing steps he will consider relevant for the generation of BCM. Moreover, he can insert optional values for all functionality’s parameters that have a default value and the output directory/filename. The order of the preprocessing steps, however, cannot be changed.

The user should also be able to log in and log out again. New users can register themselves and ask for a new password. System admins can accept new user requests, update a new password as well as activate and deactivate users.

### 3.1.1.3 Could-Haves

*“The Could-Haves are nice-to-have initiatives, as they do not quite affect the core function and would have a very small impact if left out” [49]*

The more advanced version of the application should allow the user to change the order of the preprocessing steps and to apply whatever optional parameter he wants.

The user can add a label-file which will allow a more personalised BCM generation, since the nodes can be labelled accordingly.

### 3.1.1.4 Wished-Haves

*“The Won’t-Haves are definitely not a priority for the foreseen time frame and therefore not to be included in this specific release. [49]*

A DICOM-file viewer is wished in which the input, as well as the output, can be displayed. Additionally, it should be possible that each user has his own user-space in which he can store individual files, save personalised workflows and assemble all previous BCMs.

### **3.1.2 Analysis of the advantages linked to a web application**

By now, it should be clear what needs the web application should meet and what features it will include. The advantages of a web application over the traditional FSL tool are yet to be discussed.

#### **3.1.2.1 Complexity of FSL**

The FSL application is rich and multifunctional library that various healthcare professionals use. However, to use it in its fullness a not neglectable amount of apprenticeship time is to be invested.

#### **3.1.2.2 Limited availabilities on different OS**

As for now, FSL can only be used on macOS or Linux OS. Windows users are obliged to run the application on a virtual machine. The web application is not OS-dependent and thus accessible by anyone with a stable internet connection.

#### **3.1.2.3 Lacking availability**

FSL must be installed locally on a machine. The installation process can be time-consuming, especially when not everything works as intended or when the operating system is a Windows sub-type. The web application will skip this step and the user can work with all the FSL utilities from whenever and wherever he needs to.

#### **3.1.2.4 Extensibility**

The tools offered by FSL are resourceful and vast but not complete. For some analysis steps external software must be used as a workaround. The web application will no longer be restricted to the FSL functionalities only. Additional tools - like Octave and MATLAB, to just cite two examples – can be implemented later on. These advantages make the web application more versatile and complex and leaves room for further usage.



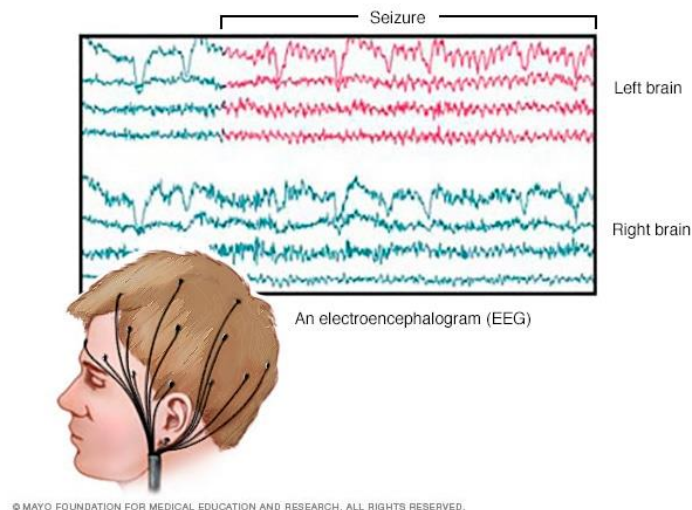
### 3.1.3 Future vision

In a final vision of this implementation medical data linked to Alzheimer is easily accessible and comprehensible for everyone. Via the application MRI-data can be uploaded and processed and their results stored in database. However, in a future vision, the application should only be the base frame for other brain analysis methods and expand on further medical tools such as electroencephalogram [50]

*An EEG is a test that measures electrical activity in the brain using small, metal discs (electrodes) attached to the scalp. Brain cells communicate via electrical impulses and are active all the time, even during asleep. This activity shows up as wavy lines on an EEG recording. [50]*

Figure 8 is an image of what an EEG could typically look like and how the electrodes are attached to the subject's brain.

Figure 8: Drawn illustration of an EEG [50]



As for now, the implementation of a brain networking analysis is the main goal of the thesis.

## 4. Implementation

Having already a broad idea of what FSL is and what it is used for, this chapter will define the conditions and set the requirements for the FSL web application implementation.

Firstly, the technical limits of the implementation will be set, starting with the application as a whole and then specifying the restrictions for the frontend, backend and the database. Once declared, the technical tools that meet these requirements are presented, briefly analysed and set within the project's implementation scope.

Next, the software design is described by disassembling the main functionality categories (Table 6) into, again, frontend, backend and database.

Finally, the testing method for each part of the application, frontend, backend and database will briefly be described.

### 4.1 Technical specifications

As for the technical requirements, they have to be SMART, meaning that they are **s**pecific, **m**easurable, **a**chievable, **r**elevant and **t**ime-bound. Therefore, they will be described in the format: *[ Part of the application ] [ will | shall | must ] [ requirement ]* [53]. Moreover, they must be non-functional requirements and align with the greater goal of this application's purpose, as stated in the introduction (Chapter 3.1.3 Future vision).

These are the requirements for the entire web applications:

- 1) The web application must be developed but not deployed.
- 2) A docker image of the application to facilitate the deployment is wished but not required.
- 3) The application must not contain external software that are not open-source and free.
- 4) The application shall be runnable on a server on which FSL and other complementary software, is already installed and set up.
- 5) The application shall only apply basic security features.

#### Frontend

The requirements for the frontend implementation are as follows:

- 1) The user interface must be user-friendly and intuitive.
- 2) The user interface is not a priority of the project, it must first and foremost be functional.

- 3) The user interfaces must be structured clear enough, so that a person with a non-medical background can use the basic features of the application and generate a BCM.

## **Backend**

The requirements for the backend implementation are as follows:

- 1) The backend must allow future extensions.
- 2) The backend shall be simply structured.

## **Database**

The requirements for the database implementation are as follows:

- 1) The database will only include compressed file paths, no files.

## **4.2 Technical implementation choices**

Deciding on the right technical choice is crucial since it can hardly be changed later. Technical choices include platform choice and, from the latter derived, choices about the applications coding language and framework. [54].

Since the platform is predefined – the goal is a *web* implementation – the following three chapters will present the language and the frameworks chosen for the FSL web application and define the naming conventions that should be used in the code implementation. For reference, according to GitHub’s Coding Best Practices, there are four main types of how variables with more than one word should be delimited. These conventions are Pascalecase, Camelcase, Snakecase and the Hungarian notation [55].

### **4.2.1 Frontend – Javascript with React**

The frontend will be written in JavaScript implementing the React library. The React library is a widespread and well-documented library that makes it easy to create interactive user interfaces. With React, entire components can be encapsulated and the DOM state managed separately. This approach allows a fast re-render of the page, which not only makes the interface more user friendly but also more reactive [56].

The user interface is not the focus of this thesis. Using a react interface will still grant a quick yet professional setup of the website’s appearance.

Table 7: Coding conventions frontend

Type	Requirements	Example
<b>React Hooks and variables related to these hooks</b>	Camelcase delimitation - must not be numbers - must not be named <i>none, null, is, else, if</i>	useReact() visibilityChair
<b>Other functions</b>	Snakecase delimitation - must not be numbers - must not be named <i>none, null, is, else, if</i>	var_to_throw_away
<b>Components</b>	Camelcase delimitation - must describe an understandable name	ReactComponent

#### 4.2.2 Backend – Python with Flask, Nypipe and Oct2Py

The backend will be written in Python. Python is a high-level, general-purpose programming language. It is known for its intuitive language, making the code easier to write and more understandable to read. Needless to say, their documentation is broad and Python is regularly ranked as one of the most popular languages [57, 58].

The FSL library itself is written in C++ and TCL – one exception being the Python installation file. However, not only can Python easily interact with the operating system via the shell, but the entire Nipype library is written in python as well. The Nypipe library is an FSL interface that lets multiple FSL processes run simultaneously and accelerates thereby the overall execution of the analysis [68]. Plus, since the application field of this web application is extremely specific and hard to understand for non-professionals of the area, it is undoubtedly useful – if not vital – that the source code can also be partly understood and read by non-professional developers [57, 58, 59].

Similarly, the Oct2Py library allows to interact with the octave application easily [60]. Notably, this is a handy extension for the generation of the brain network matrices and for eventual further expansions of the application.

Table 8: Coding conventions backend

Type	Requirements	Example
<b>Variables and functions</b>	Snakecase delimitation - lowercase and separated by an underscore - must not be a number - must not be numbers - must not be named <i>none, null, is, else, if</i>	another_variable another_function()
<b>Global variable</b>	All uppercase, with snakecase delimitation - separated by underscore	GLOBAL_VAR
<b>URL annotations</b>	Camelcase delimitation	RequestRessource

### 4.2.3 Database - PostgreSQL

The project will use a PostgreSQL database. PostgreSQL is open source and object relational-management system. It supports unstructured data types such as audios, videos or images and several advanced security features. A fast database performance is not demanded by this project since most of the data processing will be done on the server directly and immediately after presented to the user to download. Juxtaposed, the advanced security features of PostgreSQL such as data encryption will allow a safe but easy user management [61,62]. Even with the implementation of the low priority features PostgreSQL is perfectly suited to store larger MRI data.

MRI-data can be very large, especially fMRI since they are a concatenation of multiple scan volumes (Figure 4) as briefly explained in Chapter 2.1.1, which can make single DICOM fMRI file 35 MB heavy. Juxtaposed, as the Microsoft report suggest, all files that are larger than 1 MB are best stored in a filesystem [63]

Table 9: Backend - coding conventions

Type	Requirements	Example
<b>Postgres nominators</b>	All uppercase	SELECT * FROM
<b>Created variables</b>	All lowercase with Snakecas delimitation. The column names must typically use the first 3 letters of the table's name, followed by the name of the attribute, which is written in full length.	col_attribute

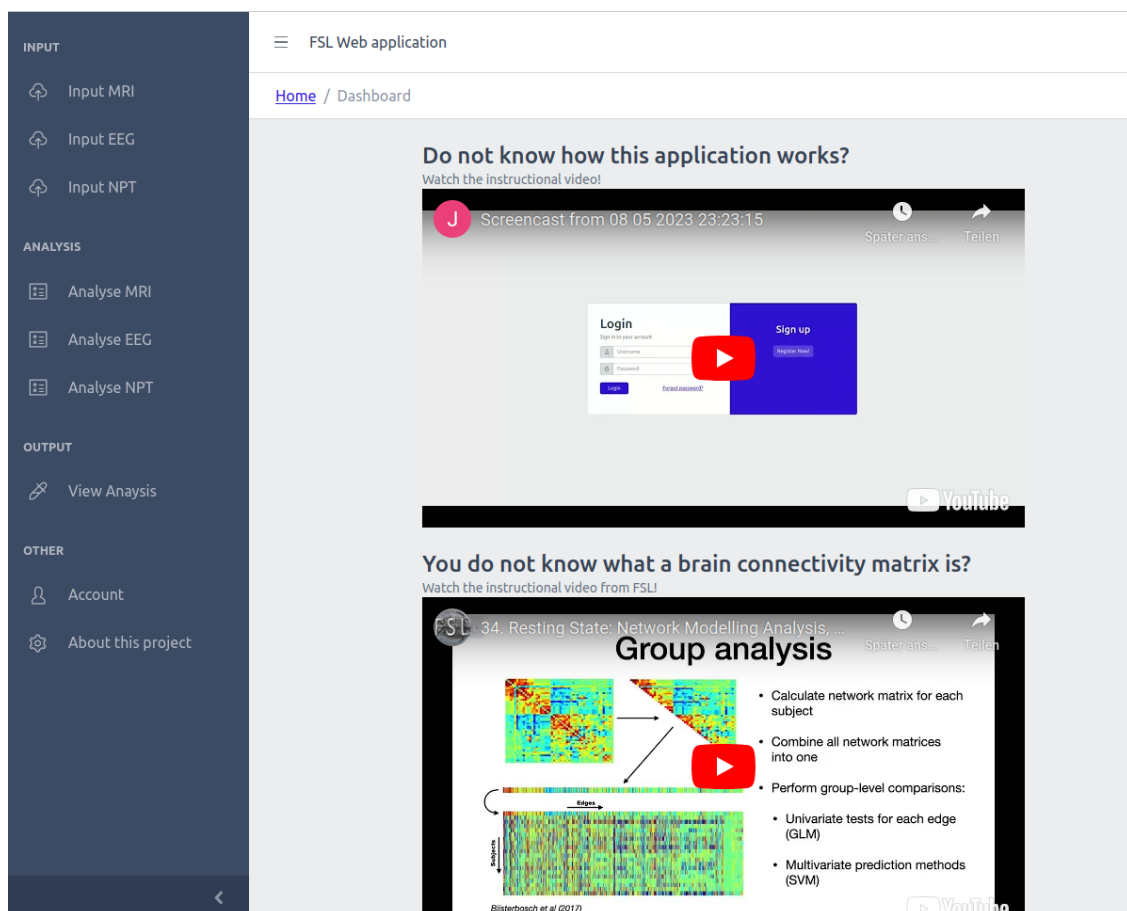
## 4.3 Software design

### Frontend

As mentioned in the beginning of this chapter, the application should create the soil for a larger application, that will englobe much more functionalities and analysis methods. Consequently, the frontend will include sections that are not developed yet. The menu bar, as displayed in Figure 9, mentions these possible future implementations.

The Frontend was built with a template from CoreUI, that can be accessed via the following link: <https://coreui.io/product/free-react-admin-template/>

Figure 9: Future functionalities



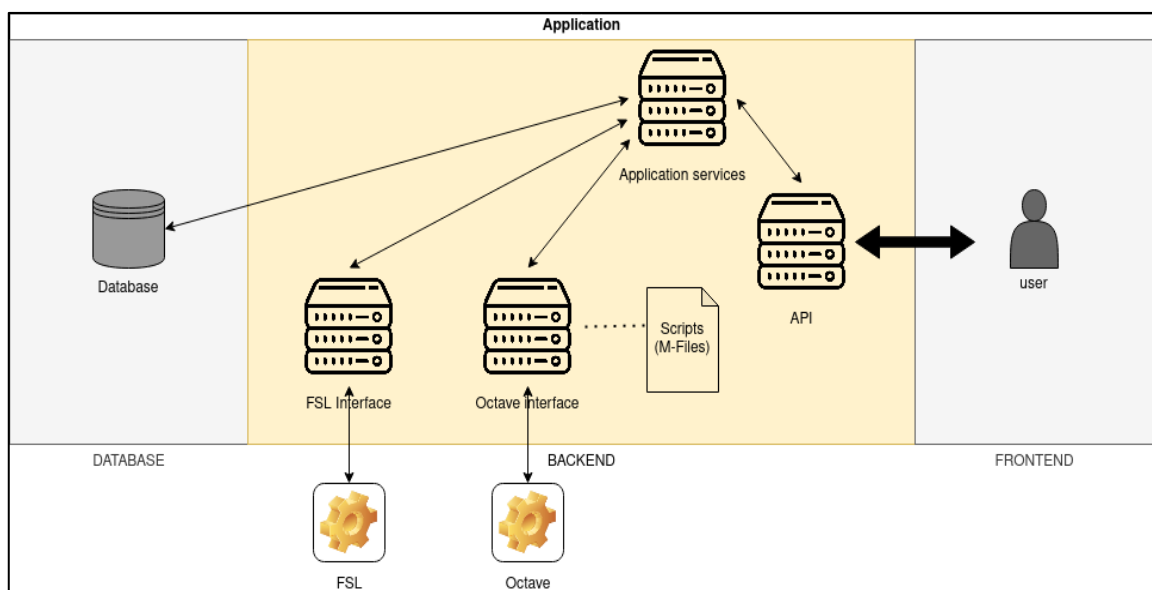
## Backend/API

The backend of this application will not only manage the project's APIs, from frontend to backend to database and vice-versa, but also manage the interaction with FSL and Octave [64]. The latter is indispensable for the creation of the BCM. As indicated in Figure 11 all user requests will always pass via the servers API-service first. There, they will be dispatched and sent to the corresponding service. For example, when the user tries to log in, the log-in data will be sent to the backend's API service. There it will send it to the corresponding application service, which in this case would be the authentication service. The authentication service will handle the transferred data and ask for identification by sending requests to the database. The database's response will be sent to the application service, which, again will forward the response back to the API. Finally, the latter will tell the frontend if the login request was successful or not.

What is more, when referring to the FSL interface, it is crucial that the FSL application is already installed properly (Appendix 1) on the server for it to execute successfully. The application will then call the Nipype library [68] that serves as FSL interface and executes the FSL commands using python. The Nipype library is thus responsible for interpreting the python commands to launch the diverse FSL processes.

Similarly, the Octave interface requires a proper Octave installation (see Appendix 1 Installation manual). The octave interface, however, can also implement external scripts and execute them. For the network analysis, the Octave interface will call the `brain_generation.m` script to generate the BCM.

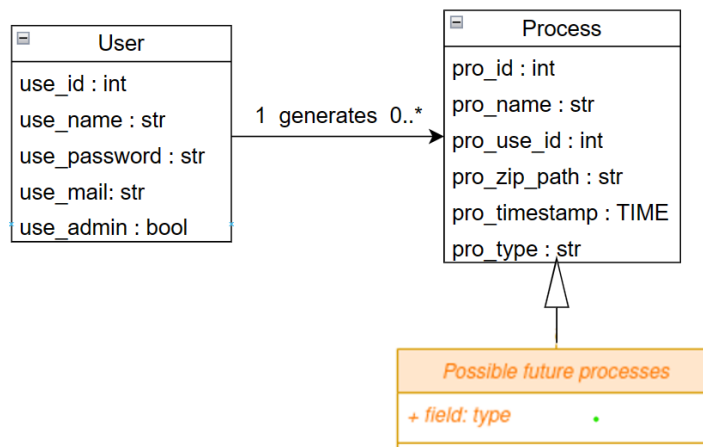
Figure 10 : Backend design



## Database

Like the frontend, the database should be adapted to a possible growth of the application. Therefore, it will rely on a rather simple structure, as seen in Figure 11. All processes generate a variety of files, regardless of the outcome. They can either success or fail. If a process failed, the “failed\_” prefix will be added to the processes type attribute (pro\_type), e.g. failed\_Matrix. It is up to the user to check if he’s process failed or succeeded. With this approach the outputs of each analysis can be stored in a “one-serves-all“-class.

Figure 11: Database model



Code 1: Creation of tables

```
1 CREATE SEQUENCE user_id_seq MINVALUE 1;
2
3 CREATE TABLE public.user (
4     use_id INTEGER NOT NULL DEFAULT EXTVAL('user_id_seq'),
5     use_name CHARACTER varying(128) NOT NULL,
6     use_password CHARACTER varying(256) NOT NULL,
7     use_mail CHARACTER varying(256) NOT NULL,
8     use_admin BOOLEAN NOT NULL DEFAULT FALSE,
9         CONSTRAINT user_pkey PRIMARY KEY (use_id),
10        CONSTRAINT uni_use_name UNIQUE (use_name),
11        CONSTRAINT uni_use_mail UNIQUE (use_mail));
12
13
14 CREATE SEQUENCE pro_id_seq MINVALUE 1;
15 CREATE TABLE public.process (
16     pro_id INTEGER NOT NULL DEFAULT NEXTVAL('pro_id_seq'),
17     pro_name CHARACTER varying(500) NOT NULL,
18     pro_use_id INTEGER NOT NULL REFERENCES public.user (use_id),
19     pro_zip CHARACTER varying(500) NOT NULL,
20     pro_timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
21     pro_type CHARACTER (100),
22        CONSTRAINT pro_pkey PRIMARY KEY (pro_id),
23        CONSTRAINT uni_pro_name UNIQUE (pro_name)
24 );
```



### 4.3.1 Analysis

The functionalities for the analysis englobe all procedures needed to generate a brain connectivity matrix. As per se, preprocessing is highly recommended to be able to create a clean BMC – it is though not required. Consequently, the generation of the BCM should be possible, with a simple fMRI input file.

#### Frontend

The GUI must allow the user to simply select an fMRI and give a desired name to this specific analysis process. The input of an sMRI mask is optional, but strongly recommended for a more accurate result. The TR used, that is fetched via the file's header – if possible – is displayed and the user can decide whether the indicated TR should be used or not [65]. Each step (1 – Node definition, 2 – Timeseries extraction and 3 – Brain connectivity matrix) of the analysis *can* be customised. As for now, the analysis should create a BCM without needing to alter any of the parameters. How a changed parameter can modify the output, will be explored in chapter 4 Outputs and tests.

Figure 12: User interface of FSL Nets

**FSL Nets Analysis**

Please select your functional MRI

Browse... sub-22\_task-flanker\_run-2\_bold.nii.gz

Use custom TR (detected TR : 2.0)

Name this process

test\_process\_22

By simply clicking on the button --Generate Connectivity Matrix-- all default parameters will be used.

**1 - Node definition** ^

Automatic estimation of nodes

Number of nodes desired: 44

Please select the mask you want to use:

none

Use BET

**2 - Timeseries extraction** v

**3 - Brain connectivity matrix** v

Generate Connectivity Matrix

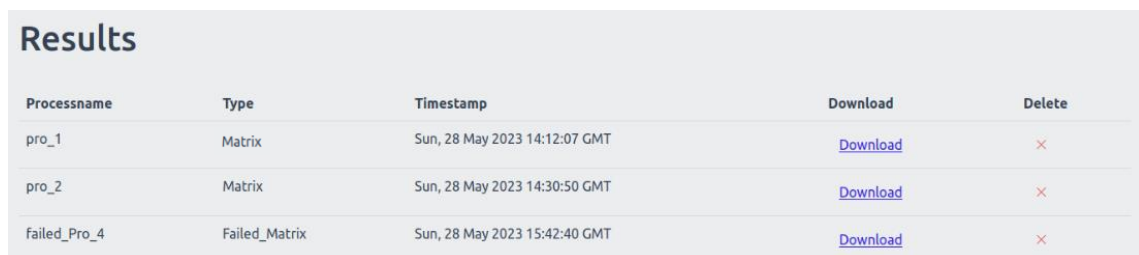
The following lines of code show how the TR is requested and fetched in the frontend.

### Code 2: Frontend - Get TR

```
1  const [fMRIFile, setfMRIFile] = useState()
2  const onfMRIFileChange = (e) => {
3    setfMRIFile(e.target.files[0])
4
5  const formData = new FormData();
6  formData.append("fMRIFile", fMRIFile);
7  fetch('http://localhost:5000/get_tr', {
8    method: 'POST',
9    body: formData})
10 .then((res) => res.json())
11 .then((data) => setTr(data.TR))
```

During the run of the analysis the *Generate Connectivity Matrix* (see Figure 12) button will be disabled and in the localstorage value of the user indicating if a process is currently being executed is set to true. As for now, running multiple processes at the same time reduces performance and puts the server at risk of crashing, which is why this restriction has been implemented. Once the analysis completed the user will be informed conformingly by a pop-up notification. The user can then navigate to the result page of the application (Figure 13), where all his processes will be listed.

Figure 13 : All processes linked to one user



Processname	Type	Timestamp	Download	Delete
pro_1	Matrix	Sun, 28 May 2023 14:12:07 GMT	<a href="#">Download</a>	×
pro_2	Matrix	Sun, 28 May 2023 14:30:50 GMT	<a href="#">Download</a>	×
Failed_Pro_4	Failed_Matrix	Sun, 28 May 2023 15:42:40 GMT	<a href="#">Download</a>	×

The user can now download the output directory in a compressed file. If the default parameters were kept, the downloaded folder should contain (see also Figure 14):

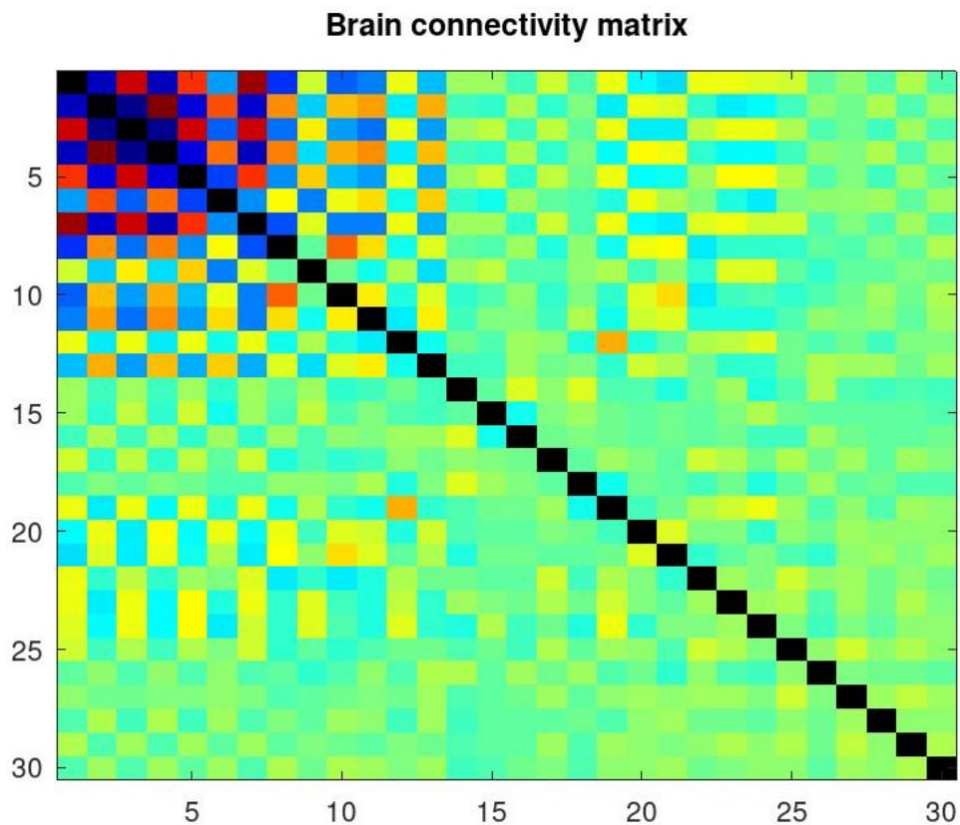
- The files generated during the melodic ICA
  - o log.txt
  - o mask.nii.gz, mean.nii.gz
  - o eigenvalues\_percent
  - o melodic\_FTmix, melodic\_ICstats, melodic\_mix, melodic\_Tmodes
  - o **melodic\_IC.nii.gz**
- The files generated during the dual regression
  - o melodic\_IC.dr (containing files depending on the number of nodes)
- The files generated during the creation of the BCM
  - o **BCM.jpg** → visual output of the BCM
  - o **BCM.csv**

Figure 14 : Result folder of a network analysis

Name	Size	Type
melodic_IC.dr	7.7 MB	Folder
Brain connectivity matrix.csv	242 bytes	CSV docum...
Brain connectivity matrix.jpg	18.2 kB	JPEG image
eigenvalues_percent	1.5 kB	unknown
fmri.nii.gz	29.7 MB	Gzip archive
log.txt	2.3 kB	plain text d...
mask.nii.gz	5.3 kB	Gzip archive
mean.nii.gz	534.2 kB	Gzip archive
melodic_FTmix	2.8 kB	unknown
melodic_IC.nii.gz	641.6 kB	Gzip archive
melodic_ICstats	215 bytes	unknown
melodic_mix	6.0 kB	unknown
melodic_Tmodes	6.0 kB	unknown
scripts+logs	4.6 kB	
dr_stage1_subject0000.txt	14.0 kB	
dr_stage2_ic0000.nii.gz	641.8 kB	
dr_stage2_ic0001.nii.gz	642.0 kB	
dr_stage2_ic0002.nii.gz	641.5 kB	
dr_stage2_ic0003.nii.gz	641.2 kB	
dr_stage2_ic0004.nii.gz	641.1 kB	
dr_stage2_ic0005.nii.gz	641.3 kB	
dr_stage2_ic0006.nii.gz	640.9 kB	
dr_stage2_ic0007.nii.gz	640.9 kB	
dr_stage2_ic0008.nii.gz	640.6 kB	
dr_stage2_ic0009.nii.gz	641.0 kB	
dr_stage2_subject0000.nii.gz	6.4 MB	
dr_stage2_subject0000_Z.nii.gz	6.4 MB	
mask.nii.gz	856 bytes	
maskALL.nii.gz	356 bytes	

Of course, the generated BCM will look different depending on the input file, Figure 15 simply serves as an example to illustrate what the output should look like.

Figure 15 : Automated BCM output

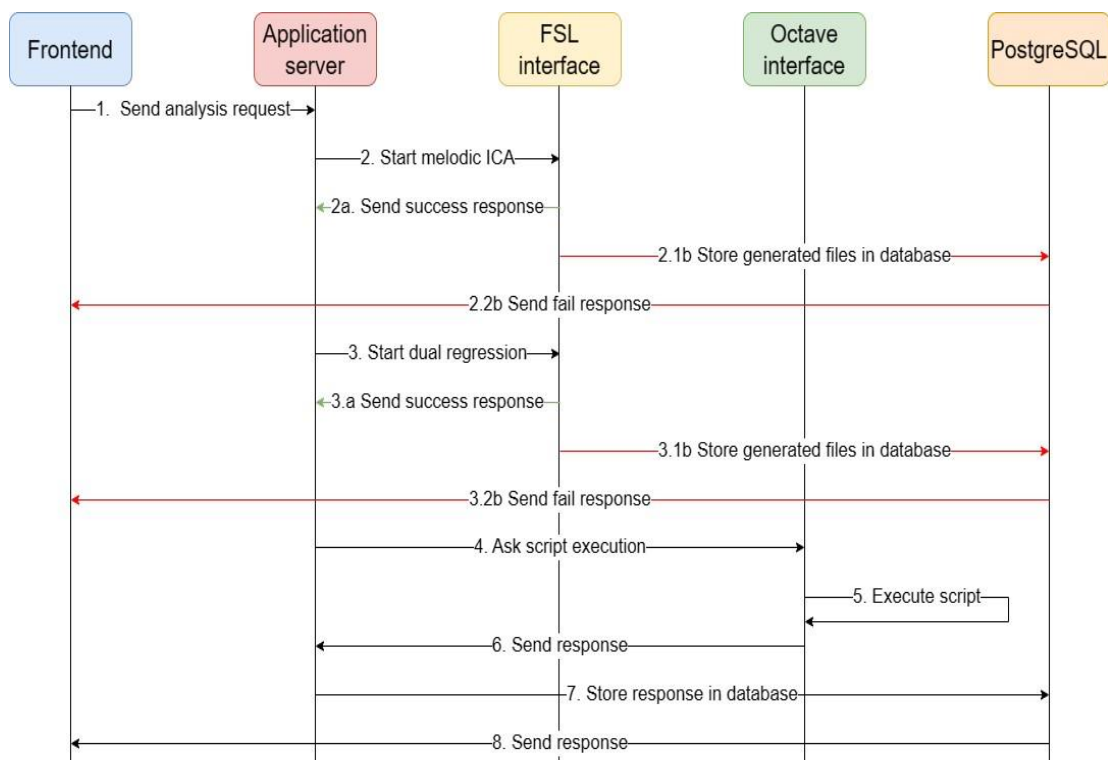


In the csv file, the exact numeric values corresponding to each field are listed.

## Backend

In the backend the process to create a BCM is triggered once the *Generate Connectivity Matrix* (see Figure 12) button is clicked. As seen in Figure 16, in the beginning, the user's request, that also carries the parameters, will be sent to the application server (step 1). The latter will then ask the FSL interface to start melodic (step 2) and then, if successfully executed (steps 2a & 3a), start the dual regression in the same way (step 3). If one of these two processes fails for whatever reasons, the files that were already generated during the precedent processes, notably the log file, are zipped and the path is saved to the database (steps 2.1b & 3.1b). A fail-response is sent via the application server to the frontend, informing the user that the process is finished, but that something went wrong (steps 2.2b & 3.2b). In a next step, the application server will ask the octave interface with the help of the oct2Py library to execute the script responsible to generate the BCM (step 4) and, if the script is available, execute it (step 5). It is the oct2Py library that creates an interface with Octave and thereby allows the script, that is responsible for the creation of the BCM, to be accessed. The octave interface informs the application if the execution of the script was successful or not (step 6) and stores the generated data in the database (step 7). If step 5 did not succeed, the keyword "failed\_" will be prefixed to the *processing\_type* attribute (see Figure 11, chapter 4.3 [Software design](#) – Database). Finally, the user is informed that the process has been run and he can view and download the output in the result page (step 8).

Figure 16: Backend creation of a BMC sequence diagram



From the octave interface the corresponding script is called in the following way. The octave command, *createMatrix* must correspond to the file name containing the script.

### Code 3: Backend - Octave interface

```
1     def add_octave(melodic_filepath: str,
2                   tr: float,
3                   normalisation: bool,
4                   matrixType: str,
5                   colour: str,
6                   matrixTitle: str):
7         oct = octave.Oct2Py()
8         oct.addpath(app.config["FSL_PATH"])
9         oct.addpath(app.config["OCTAVE_PATH"])
10        oct.addpath('../fsl_backend/scripts')
11        try:
12            oct.createMatrix(melodic_filepath, tr,
13                             normalisation, matrixType, colour, matrixTitle)
14            oct.exit()
15        except Exception as e:
16            print(str(e))
```

In the above Code 2, lines 7 to 10 will make sure that the application has access to all FSLNets scripts, to the octave library and the web application's own scripts. Then in line 12 the script which will calculate the BCM is called. Oct2Py will associate the script's filename to the function name.

### Code 4: octave script, named createMatrix.m

```
1 % Parameters :
2 % melodic_filepath: string | Example: '../X_PROCESS/me_IC.dr'
3 % tr: number | Threshold, example: 0.42
4 % normalisation: boolean | Example: 1
5 % matrixType: string | Example: ['corr']
6 % colout: String | Example: ["jet"]
7
8 function createMatrix(melodic_filepath,
9                       tr,
10                      normalisation,
11                      matrixType, colour, matrixTitle)
12     ts = nets_load(melodic_filepath, tr, normalisation)
13     net_matrix = nets_netmats(ts,1,matrixType)
14     corr=reshape(net_matrix,ts.Nnodes,ts.Nnodes)
15     corr(corr==0)=NA
16     imagesc(corr)
17     colormap(colour)
18     title(matrixTitle)
19     close all force
20 end
```

This is the core of the application, which is why this file will be explained in more detail. The function *createMatrix* takes 6 parameters, of which only the first two are indispensably required. The path to the melodic directory (*melodic\_filepath: me\_IC.dr*)

and the TR of the current fMRI file. In line 12, all melodic files from the melodic directory are loaded. Then, in line 13 a first matrix is generated and then properly reshaped in line 14. In line 15, all values of the diagonal are set to NA (null) in order to avoid ambiguous coloration of the matrix and distinguish the diagonal line clearly from the rest of the matrix. Eventually, in line 16 to 18 the BCM is named, coloured and labelled after the user's preference.

With the oct2Py interface other octave scripts can extremely easily be added later on. For example, if a *wishedAnalysis* script is available, the latter file just needs to be added to the application's folder and then called via the octave interface with the command `oct.wishedAnalysis([parameters])` (see Code 2, line 12).

### **Database**

The tables used for a successful implementation of the MRI-Analysis functions into the database are the tables; Process, User (see Figure 11, chapter 4.3 [Software design – Database](#)).

## 4.3.2 Preprocessing

Thanks to the Nipype library the implementation of the diverse preprocessing steps is relatively similar, independently of what type of preprocessing is being applied. The main difference will be the parameters and the associated output. Therefore, the preprocessing will only briefly be described.

### Frontend

Under the input section, the user will have the choice of what type of MRI he wants to process, anatomical or functional. Each preprocessing steps has its own parameters and default values that can be changed after the user's preference. Changing the order of preprocessing with a simple drag&drop is not yet possible.

Figure 17: GUI Preprocessing

The frontend must send all parameters in the correct form, so the application service can process it correctly. These parameters are all in a React form in which they will be filtered by their belonging to a certain preprocessing step.

### Code 5: Frontend - Initial form

```
1  const initialForm = {bet:false, mcf:false, sli:false, spat:false};
2  const [form, setForm] = React.useState(initialForm);
```

All BET parameters are, therefore, named with a prefix “bet\_”, all Motion correction parameters with the prefix “mcf\_” (for FSL McFlirt) and all Slice timing correction parameters with the prefix “sli\_”. Additional preprocessing steps can be added if wished.

#### Code 6: Frontend - Parameters

```

1  function extractParams(allParams, steps) {
2    let extracted = {}
3    for (let s of steps) {
4      if (allParams[s] == true) {
5        let typeParams = {}
6        for (let p of allParams.target.elements) {
7          if (p.name.substring(0, 3) == s) {
8            Object.assign(typeParams,
9              {[p.name.substring(4, p.name.length)]:p.value})}
10         typeParams N= setParams(typeParams, s)
11         Object.assign(extracted, {[s]:typeParams})}
12     }
13     return extracted
14   }
15   function setParams(extractedParams, type){
16     if (type == "bet"){
17       return setBetParams(extractedParams["bet"])
18     }else if (type == "mcf"){
19       return setMcfParams(extractedParams["mcf"])
20     }else if (type == "sli"){
21       return setSliParams(extractedParams["sli"])
22     }
23   }
24 }

```

Once the form gets submitted via the *Start Preprocessing* button all parameters are assembled, adjusted and sent to the backend with the correct formatting (dictionary).

#### Code 7: Frontend - Parameter submission

```

1  const startPreprocessing = (e) => {
2    e.preventDefault();
3    alert("Your data is being preprocessed. This may take a while...")
4    const extractedParams = extractParams(form);
5    if (form.bet == true){form["bet"] =
6      setBetParams(extractedParams["bet"])
7    }
8    if (form.mcf == true){form["mcf"] =
9      setMcfParams(extractedParams["mcf"])
10   }
11   if (form.sli == true){form["sli"] =
12     setSliParams(extractedParams["sli"])
13   }
14   /*-- Other submit actions irrelevant for the parameters--*/
15   const formData = new FormData();
16   formData.append("file", file);
17   formData.append("processName", processName);
18   formData.append("bet", JSON.stringify(form.bet));
19   formData.append("mcf", JSON.stringify(form.mcf));
20   formData.append("sli", JSON.stringify(form.sli));
21   fetch('http://localhost:5000/preprocess', {
22     method: 'POST',
23     body: formData})
24     .then((res) => res.json())
25     .then((data) => { /* Re-initialising form and window--*/ })
26     .catch((err) => console.log(err))

```

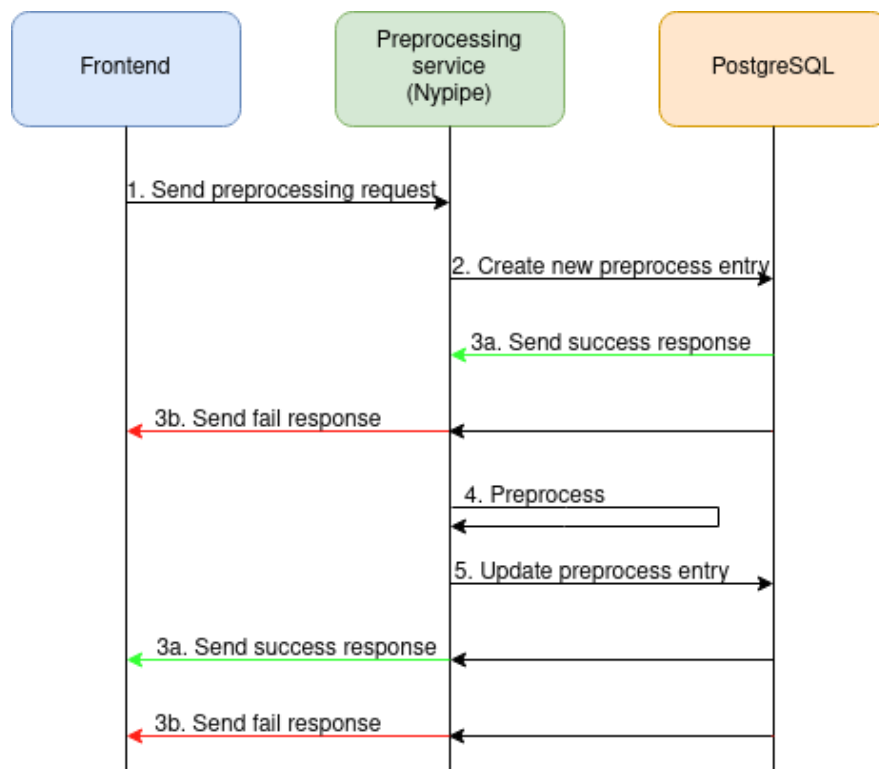


## Backend

Similarly to the network connectivity analysis, the user will send its request to the FSL interface Nypipe (step 1). There it will make sure that the folder location in the server is given. This is important since the preprocessing steps can generate a lot of different files (step 2). Then, if every process run successfully (step 3a) the data will be preprocessed (step 4), the files compressed and the process saved into the database (step 5). Finally, the user receives a notification indicating that the preprocessing has finished and that it can be viewed. If the preprocessing failed, the user will be informed correspondingly and the keyword “failed\_” will be prefixed to the *processing\_type* attribute (3b).

The parameters needed for the preprocessing steps are included in the preprocessing request (step 1).

Figure 18: Sequence diagram preprocessing



The FSL interface Nypipe must be able to process all parameters, if given or not, which can be achieved with the python `**kwargs` functionality. In the next section of Code 8 the FSL interface which handles the BET requests is shown. The parameters are classified into compulsory parameters, without which the preprocessing step cannot be executed, default parameters, these parameters provide a default value, even if it was not given by the user, and the optional parameters that will, if not indicated, just be ignored.

It has to be noted that all the preprocessing services proposed by Nypipe are built in the same way, only the name of the function changes. Therefore, this example also serves as model for all other preprocessing steps that are, or are yet to be implemented.

### Code 8: Backend - Preprocessing

```
1 def launch_bet(input_file, **kwargs):
2     myb = fsl.BET()
3     ##### MANDATORY PARAMETERS #####
4     myb.inputs.in_file = input_file
5     myb.inputs.out_file = set_extraced_brain_name(input_file, "_brain")
6     ##### OPTIONAL PARAMETERS WITH DEFAULT #####
7     myb.inputs.frac = kwargs.get("frac", 0.5)
8     myb.inputs.vertical_gradient = kwargs.get("vertical", 0)
9     ##### OPTIONAL PARAMETERS WITHOUT DEFAULT #####
10    if kwargs.get("outline"): myb.inputs.outline = kwargs.get("outline")
11    if kwargs.get("mask"): myb.inputs.mask = kwargs.get("mask")
12    if kwargs.get("no"): myb.inputs.no_output = kwargs.get("no")
13    if kwargs.get("radius"): myb.inputs.radius = kwargs.get("radius")
14    if kwargs.get("center"): myb.inputs.center = kwargs.get("center")
15    if kwargs.get("thresh"): myb.inputs.threshold = kwargs.get("thresh")
16    if kwargs.get("mesh"): myb.inputs.mesh = kwargs.get("mesh")
17    if kwargs.get("skull"): myb.inputs.skull = kwargs.get("skull")
18
19    match kwargs.get("vars"):
20        case "robust":
21            myb.inputs.robust = True
22        case "remove_eyes":
23            myb.inputs.remove_eyes = True
24        case "reduce_bias":
25            myb.inputs.reduce_bias = True
26        case "padding":
27            myb.inputs.padding = True
28        case "surfaces":
29            myb.inputs.surfaces = True
30
31    if kwargs.get("t2_guided"):
32        try:
33            myb.inputs.t2_guided = kwargs.get("t2_guided")
34        except Exception as e:
35            print(e)
36        try:
37            myb.run()
38        except Exception as e:
39            print(e)
```

### **Database**

The tables used for a successful implementation of the Preprocessing functions in the database are the tables; Process, User (see Figure 11, chapter 4.3 [Software design](#) – Database).

All MRI files (input and output) are saved on the server directly. The database will only reference the path to the file's location on the server.

### 4.3.3 User management

Beside the core functionalities that are directly linked to the FSL services, the web application also integrates user-oriented functionalities, which are (see table 3, chapter 3.1 [Functionalities](#) ) :

- Login and logout function
- User management
  - o User management for regular users (registration, new password request)
  - o User management for system admins (activate and deactivate users)

The user management is split into two sub-sections. On the one hand, there are the functionalities for a non-admin user and on the other hand, the system administrator will be able to accept new registrations request and block users.

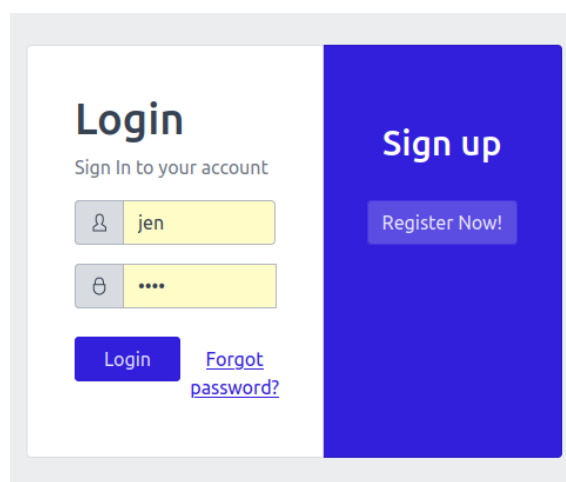
#### 4.3.3.1 Login and Logout

For the login and logout management a very simple micro-service-oriented approach was chosen which is explained as follows.

##### **Frontend**

The Login GUI is very conventionally designed after a template accessible via the following link: <https://coreui.io/product/free-react-admin-template/>

Figure 19: GUI Login



## Backend/API

The authentication service runs in two steps, as indicated in Figure 20 and Figure 21. Firstly, the user is identified and, secondly, the thereby generated Token verified to validate the session, after an example of Shane Larson.

Figure 20: Backend - Verification of credentials [67]

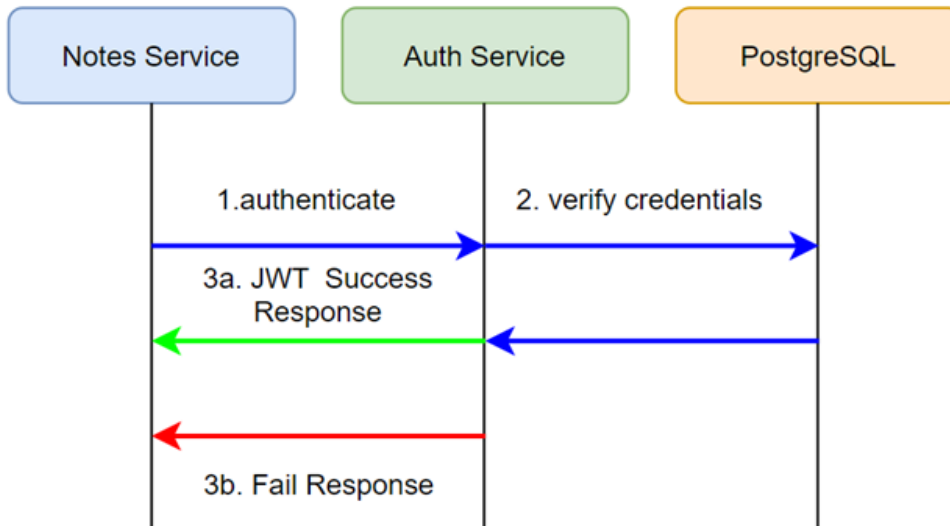
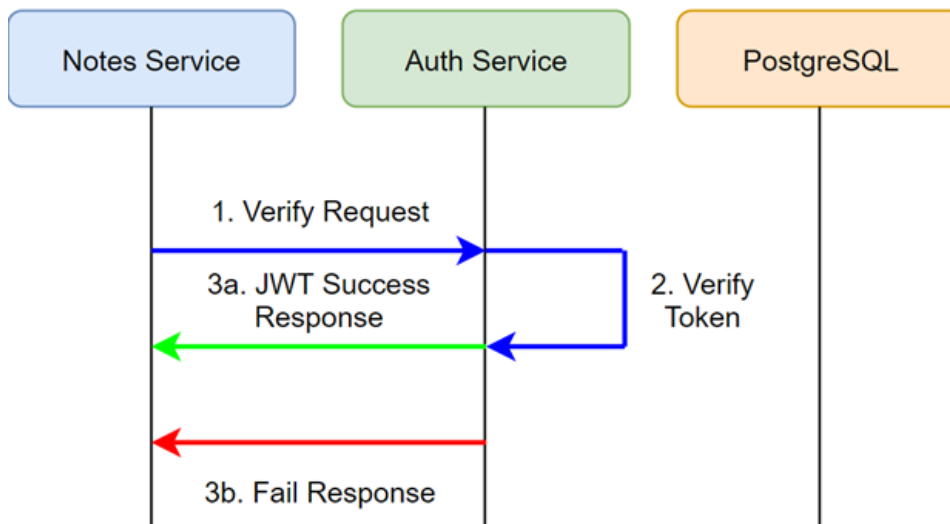


Figure 21: Backend - Verification of the session token [67]



The received credentials secrets (passwords) are hashed with the Secure Hash Algorithm 2 (SHA-2) immediately before sending the request to the database. Currently even some cryptocurrencies make use of this hashing method for transaction verification for example [66]. The token, as well as the username are saved in the user's localstorage.

### Code 9: Backend - Password encryption

```
1 hash_object =
2     hashlib.sha256(bytes(client_secret_input, 'utf-8'))
3 hashed_client_secret = hash_object.hexdigest()
4 authentication_check =
5     db_user.authenticate(client_id, hashed_client_secret)
```

### Code 10: Backend - Recieving encoded password from database

```
1 try:
2     encoded_jwt = jwt.encode(payload.__dict__, AUTHSECRET,
3                             algorithm='HS256')
4     resp = ResponseAuth(encoded_jwt, EXPIRESSECONDS, isAdmin)
5     c.close()
6     cnx_db.close_commit_db(c, cnx)
7     return resp.__dict__
```

#### Database

The tables used for a successful implementation of the Login and Logout functions into the database are the tables; User (see Figure 11, chapter 4.3 [Software design – Database](#)).

Moreover, when sending a simple select query to the User table, it can be noticed that the password outputs are hashed.

### Code 11: Database - Hashed password

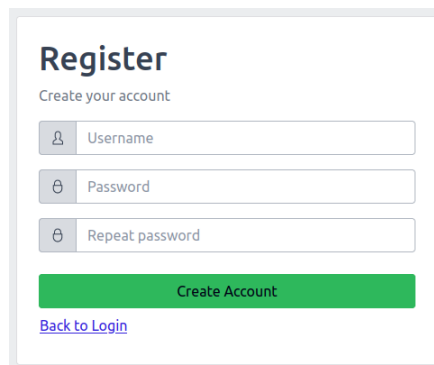
```
fsl=# SELECT * FROM public.user;
 use_id | use_name | use_password | use_admin
-----+-----+-----+-----
    12 | jen     | 9d4e1e23bd5b727046a9e3b4b7db57bd8d6ee684 | t
```

#### 4.3.3.2 Register

##### Frontend

Again, the register GUI is conventionally designed after the template accessible via the following link: <https://coreui.io/product/free-react-admin-template/>

Figure 22: Frontend Registration



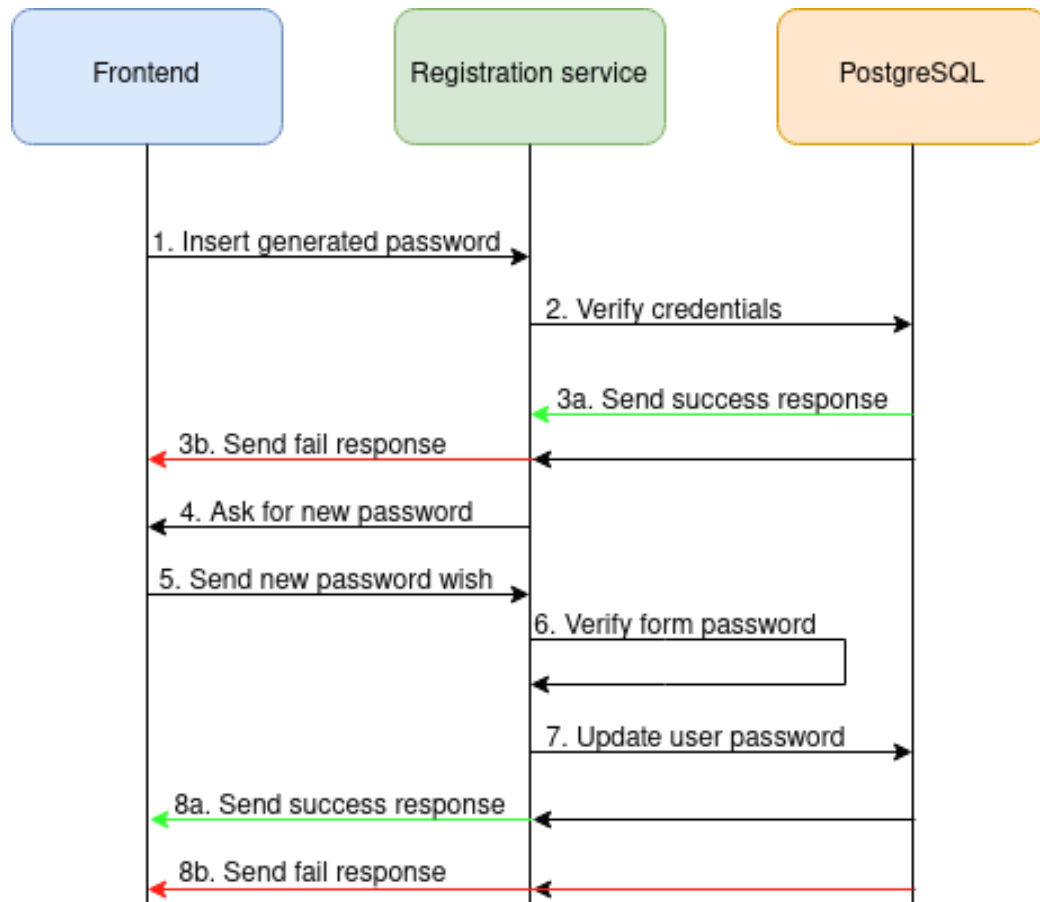
The screenshot shows a registration form with the following elements:

- Title: Register
- Subtitle: Create your account
- Input fields: Username, Password, Repeat password
- Submit button: Create Account (green)
- Link: Back to Login (blue)

## Backend/API

As seen in the below Figure 23, once the user wants to create a new user profile, the username, mail address and password are sent to the database to verify that the user is not created twice, either with the username or with the mail address.

Figure 23: Registration service



## Database

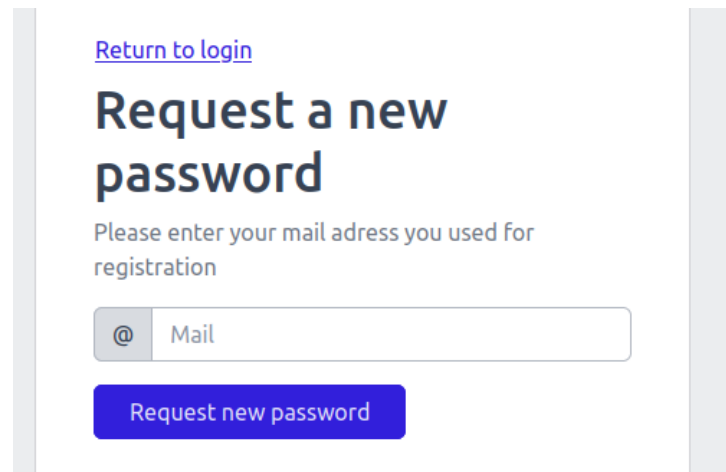
As we can see in the creation of the database, Code 1 Creation of tables, it should not be possible to create a duplicate user, as the user ID is generated dynamically and neither the user id, user name nor the user mail is allowed to be empty, which is indicated by the keyword NOT NULL.

The tables used for a successful implementation of the User management functions into the database are the tables; User (see Figure 11, chapter 4.3 [Software design](#) – Database).

### 4.3.3.3 Request new password

It can happen, that the user forgets his password. In this case he can request a password reset. Figure 24 displays what the user will see once he requests a new password.

Figure 24: Frontend, forgot password



[Return to login](#)

## Request a new password

Please enter your mail adress you used for registration

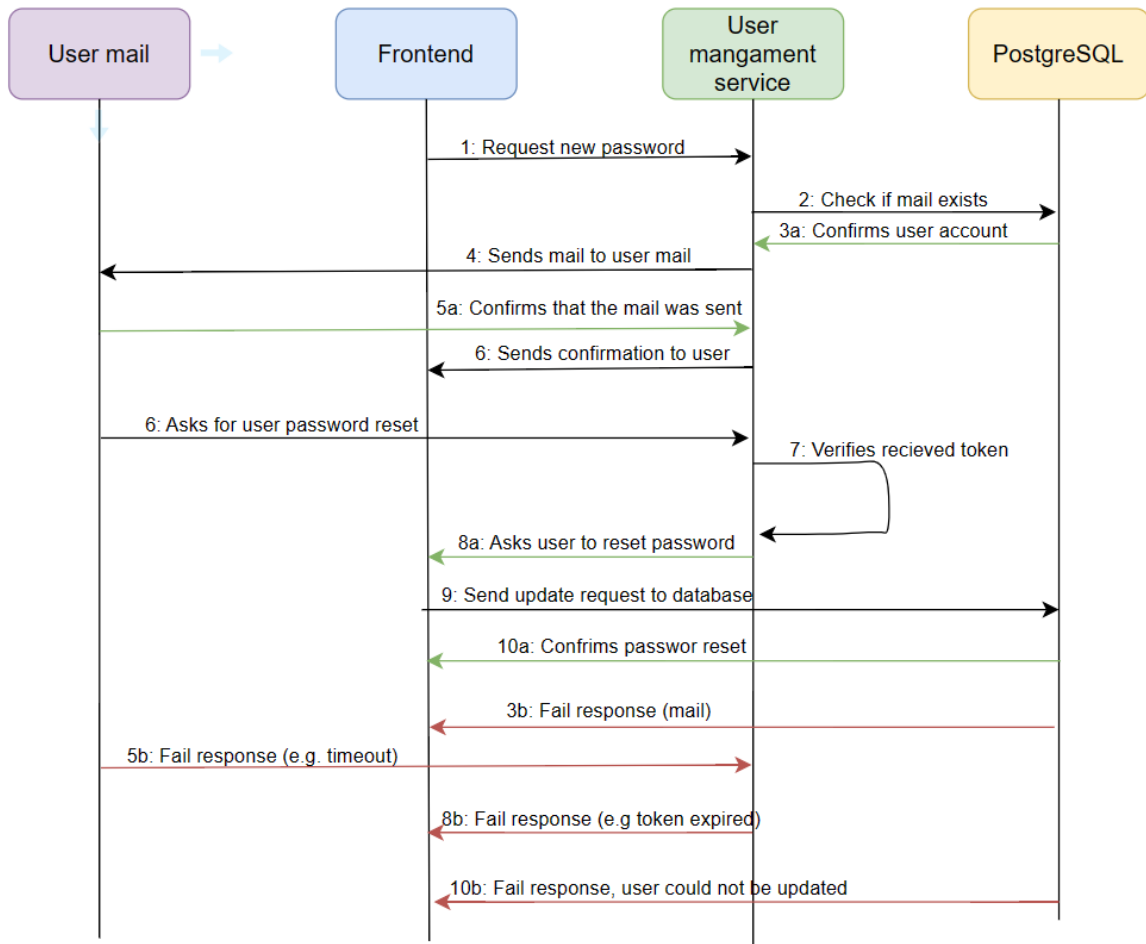
Request new password

When requiring the new password, it is important, that the mail address typed in (Figure 24) corresponds to the mail address the user provided during registration. The submitted mail address will be verified in the backend, using the following code to access the database.

Code 12: Retrieving the user's mail from database with python

```
1 def get_userMail(user_mail):
2     query = "SELECT * FROM public.user WHERE use_mail = %s"
3     cnx = cnx_db.open_db()
4     if cnx is not None:
5         c = cnx.cursor()
6         try:
7             c.execute(query, (user_mail,))
8             if c.rowcount == 1:
9                 return 1
10            else:
11                return 0
12        finally:
13            c.close()
14            cnx_db.close_commit_db(c, cnx)
15    else:
16        raise Exception("No connection!")
```

Figure 25: Password reset request



The password reset procedure is set off when the user clicks on the “Forgot password?” link in the Login page (Step 1). If the user mail is in the database (Step 2) the user management service will send a mail to the user’s mail address, with a link that includes a newly generated Token (Step 3,4). If the mail was successfully sent (Step 5), a pop up will appear that asks the user to follow the instructions sent to personal mail account (Step 6). The user will then be redirected to the login page. Once the user opens the mail and clicks on the link, a verification request of the Token will be sent to the User management service, which will – if the Token is valid (Step7) – navigate the user to the “Password reset” page (Step 8). The user will then be requested to reset his password (Step 9). If the password is not empty, the user’s password is updated in the database and the user will receive a confirmation pop-up (Step 10).



## 4.4 Tests and output

Even though it can be challenging for a user without any medical background knowledge to state if the extracted BCM or the MRI result looks satisfying, the code quality and coherence can still be tested. Testing code is indispensable if the application should be robust and have a minimum resistance not to bug on arbitrary user exploration. There are usually three categories of testing: functional testing, non-functional/performance testing and maintenance testing [69]. As for this project, the tests limit themselves to unit and integration testing only.

### 4.4.1 Testing requirements

#### Frontend

For the react frontend, the Jest library will be used. It is currently the most popular testing framework. Meta, formerly Facebook, takes care of the maintenance [70]. Moreover, a separate folder for each page component is set up for further testing implementations.

To run the tests, run the following commands in the command line terminal: `npm jest`. For further run options, please consult the Jest Documentation: <https://jestjs.io>.

A simple example is to check if the loading button is disabled during the process of a matrix generation.

With the jest library the component to be tested is copied using the keyword

```
describe(Component, () => {
```

In the parenthesis, the developers test will be written. Then with the *expect* keyword the expected results will be compared to the actual result. If both results are equal, the test returns true, otherwise, the test will return an error.

```
expect(ReactStatus of variable).toEqual(The result I expect from this test)}.
```

Since the frontend of the application was built on a template, these templates will not be tested.

## Backend

For the flask – python backend, the pytest library is used. An integrated testing feature in Python, called unittest, can be considered as equally powerful as pytest. The latter framework was chosen due to the more detailed output [71] and the personal preference of the author.

All files that want to be included in the pytest test run must have either “test\_” as a prefix or “\_test” as a suffix to the file’s name. Then, the tests can simply be run by sending the command `pytest <directory of test files>`. If warnings about deprecated packages should be ignored, the parameter `pytest -W ignore::DeprecationWarning` can be added to the command. For further run options, please consult the [pytest Documentation: https://docs.pytest.org/en/7.3.x/contents.html/](https://docs.pytest.org/en/7.3.x/contents.html/).

Like for the frontend, a separate folder for all tests has been prepared. One example is to check if the file uploaded by the user corresponds to the file that is retrieved by the Nypipe interface. A simple test will look as follows:

### Code 13: Set up of a correct test

```
1 def test_files(set_vars):
2     testfile = "/home/jen/X_DATA/structural.nii.gz"
3     myb = fsl.BET()
4     myb.inputs.in_file = testfile
5     assert myb.inputs.in_file == testfile, "Bet files do not
match"+testfile+" - "+myb.inputs.in_file
```

Since the file is the same, the result should be correct.

### Figure 26: Successful backend pytest test

```
..... test session starts .....
platform linux -- Python 3.10.6, pytest-7.3.1, pluggy-1.0.0
rootdir: /home/jen/PycharmProjects/predprodaLzhesmer-scratch/fsl_frontend/fsl_backend/tests
collected 3 items

test_fsl_bet.py ...s [100%]

..... 2 passed, 1 skipped in 2.08s .....
```

However, if the code is slightly changed, an error will appear:

### Code 14: Set up of an incorrect test

```
1 def test_files(set_vars):
2     testfile = "/home/jen/X_DATA/structural.nii.gz"
3     myb = fsl.BET()
4     myb.inputs.in_file = testfile
5     assert myb.inputs.in_file == testfile, "Bet files do not
match"+testfile+" - "+myb.inputs.in_file
```

Figure 27: Failed backend pytest test

```
> raise TraitError(
    object, name, self.full_info(object, name, value), value
)
E traits.trait_errors.TraitError: The 'in_file' trait of a BETInputSpec instance must be a pathlike object or string representing an existing file, but a value of '/home/jen/X_DATA/functional.nii.gz' <class 'str'> was specified.

../../../../env/lib/python3.10/site-packages/traits/base_trait_handler.py:74: TraitError
or
===== short test summary info =====
FAILED test_fsl_bet.py::test_files - traits.trait_errors.TraitError: The 'in_file' trait of a BETInputSpec insta...
===== 1 failed, 1 passed, 1 skipped in 2.19s =====
```

This test seems trivial at first sight, but in the case of a missing letter in the file name for example, the error can quickly be traced back to and thus make maintenance work more efficient in the long run.

## PostgreSQL

Due to the size of the database (two tables only) no testing process was yet implemented for the database.

### 4.4.2 Outputs and results

By now it is clear what needs the web application fulfils and how the functionalities are implemented technically. Finally, the BCMS generated by the application must be verified to their correctness regarding the parameters that the user can modify or ignore.

As stated beforehand, the tests described in the following few paragraphs aim to verify, when a parameter is isolated and then modified, if these modifications are reflected in the output. Moreover, only the parameters that can be checked without the need of any medical background information will be tested. All parameters that can be set by the user himself are listed in Table 28 below. Other tests such as user input formats will not be handled in this paper.

Figure 28 : BCM parameters

N	Parameter	Input	Mandatory	Comment
1	Input file, fMRI file for BCM	File	Yes	If no file is selected, the button to launch the process is disabled. Similarly, if the file selected does not correspond to the expected file extension, the button to launch the process rests disabled as well.
2	TR used	Float	Yes, default: DICOM header	If the TR is readable in the fMRI's header file, it will retrieve the TR from the header directly. Otherwise, the user must indicate the correct TR manually.

3	Process name	String	Yes	If the process name was already used by the user, the button to launch the process is disabled. The process name will be the name of the output zip file.
4	Nb of nodes	Integer	No, default: automatic estimation	The nodes can be estimated automatically by melodic. The number of nodes will define the size of the BCM ( $nbNodes * nbNodes$ ).
5	Mask	File	No, default: none	The mask file must correspond to the input fMRI file. Otherwise, the melodic process will fail.
6	Bet	Boolean	No, default: No	The basic brain extraction procedure is applied to the fMRI input file.
7	Normalise timecourse	Integer	No, default: No	This input should not be modified.
9	Normalisation	Boolean	No, default: True	This input should not be modified.
10	Correlation	String	No, default: Full correlation	FSLNets gives the user the possibility to create a network using different approaches. Each string corresponds to one of these approach-options, that are; covariance, full correlation, partial correlation, amplitudes only, Ridge regression (partial), Hyvarinen
11	R to Z	Integer	1, default: 1	If 0, this options is turned off.
12	Matrix' name	String	No, default, "Brain connectivity matrix"	This input will be the title of the BCM image generated.

### Tests: mandatory parameters

As mentioned in the introduction the goal of the application is to allow a regular user (user without any medical background) to create a BMC via a web application. Therefore, when the user only fills in the mandatory input fields, the application should still create a plausible BCM.

For parameters 1 and 2 the application will not test the correctness of the filetype or of the TR, simply because these two parameters must be gathered during the MRI sessions. However, the output BCM should be different if the input fMRI file changes. The two figures on the next page (Figure 29, 30) are the outputs of three different fMRI input files.

Figure 29 : BCM A, changing parameters 1 and 2

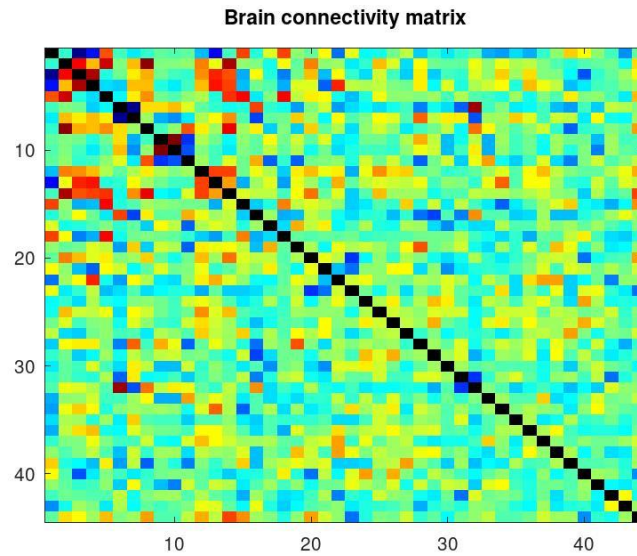
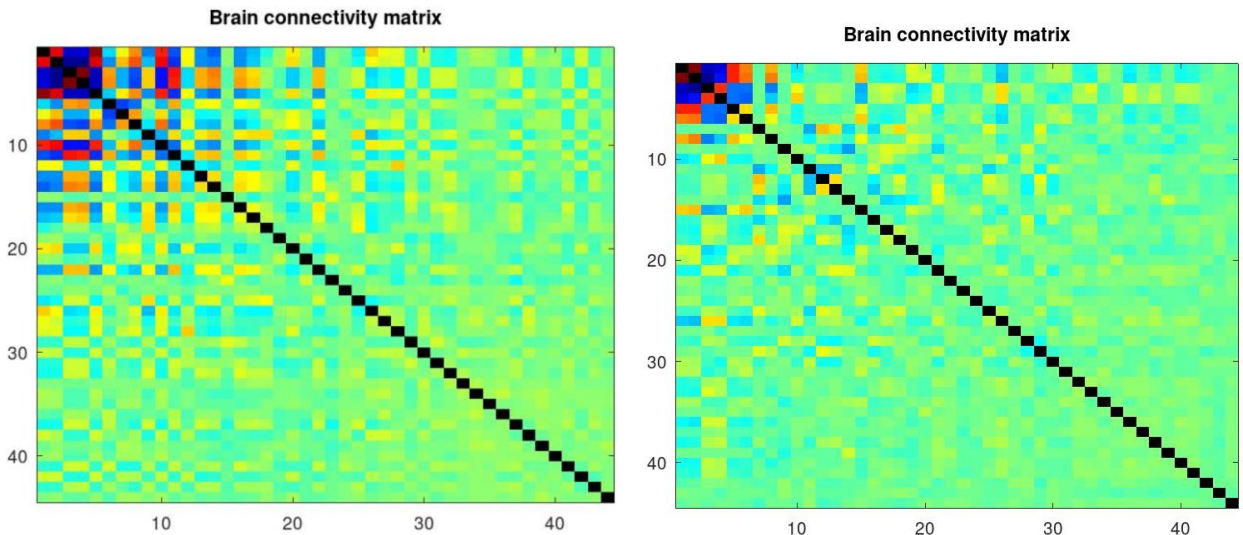


Figure 30 : BCM B and C, changing parameters 1 and 2



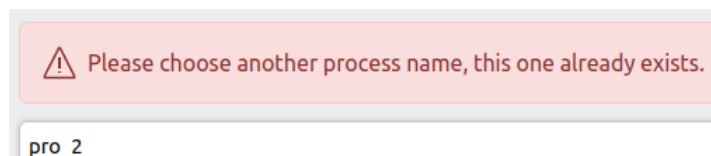
It must be noted, that each BCM is mirrored on the diagonal.

The application itself will only verify if the process name (parameter 3) already exists in the database and advise the user to change the process name.

Figure 31 : Process' names in database

pro_id	pro_name	pro_use_id	pro_zip	pro_timestamp
1	pro_1	1	/home/jen/FSL/1/MATRICES/pro_1/pro_1.zip	2023-05-28 14:12:07.821991
2	pro_2	1	/home/jen/FSL/1/MATRICES/pro_2/pro_2.zip	2023-05-28 14:30:50.919806

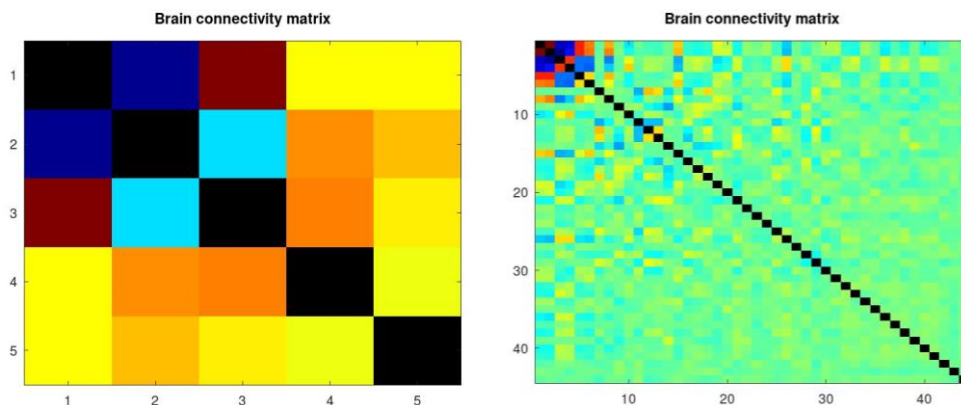
Figure 32 : Invalid process name (not unique)



### Output: parameter 4

The ROIs can be estimated automatically by the application it is, however, recommended to only use about 30 regions with the ICA method. It must consequently be tested if there is a difference between the BCMs generated when all other parameters stay the same and only parameter 4 is modified. Conforming to the description in Table 28, if the BCM is generated setting parameter 4 on number 5, the expected outcome matrix should have a size of 25, respectively 1'936 if set to 44. As it becomes evident in Figure 33, the output corresponds to the expected result.

Figure 33 : BCM with 5 nodes (right) and 44 nodes (left)



### Outputs: parameters 10 and 12

FSLNets provides six options on how a matrix can be generated. However, only the options, “full correlation” (Figure 34), “partial correlation” (Figure 35), “partial correlation” “full correlation with r to z” (Figure 36), “covariance” (Figure 37) and “partial ridge regression” (Figure 38) can be generated. For better readability, in the following outputs the csv file has been overlaid to the BCM image.

Figure 34 : BCM - full correlation

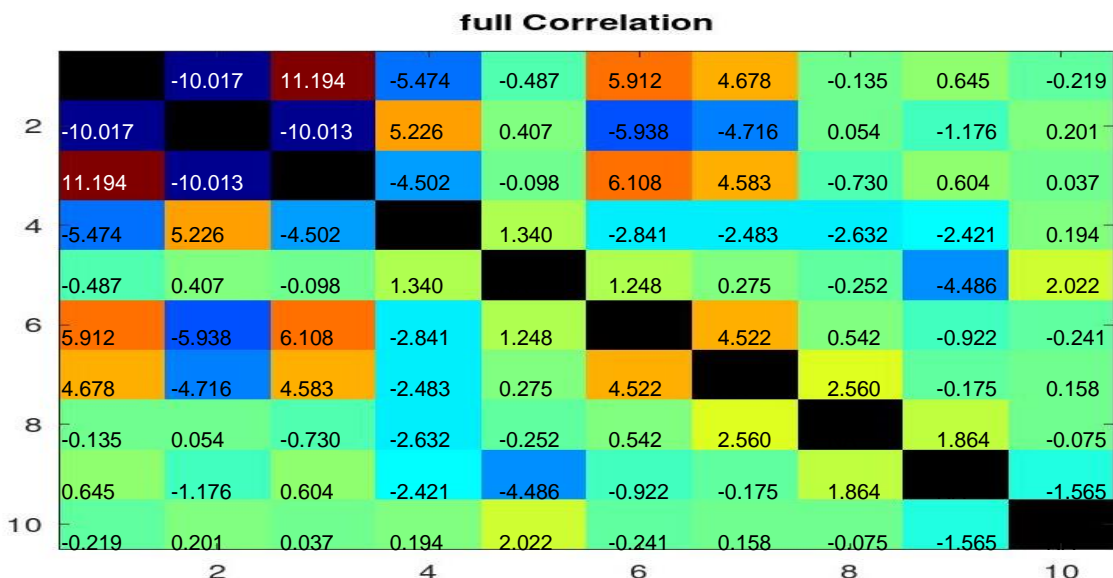


Figure 35 : BCM - Partial correlation

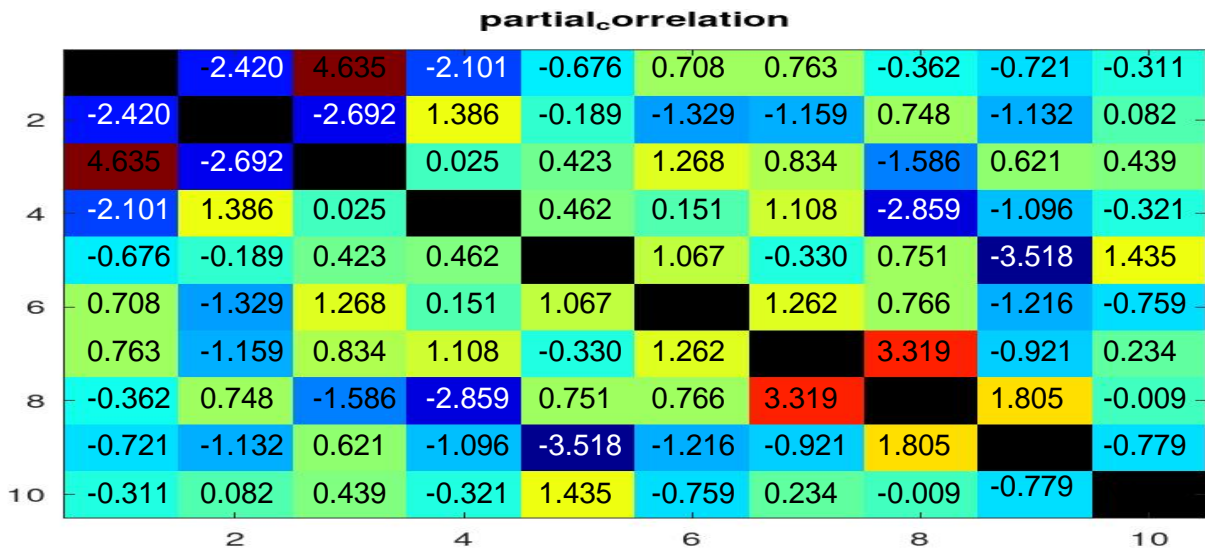


Figure 36 : BCM - full correlation, with r to z

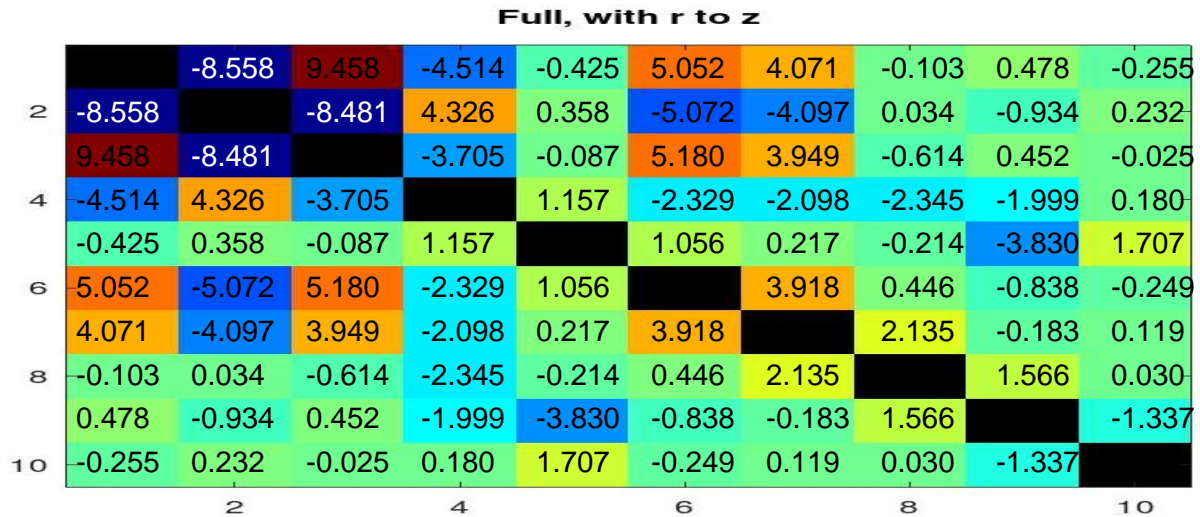


Figure 37 : BCM - Covariance

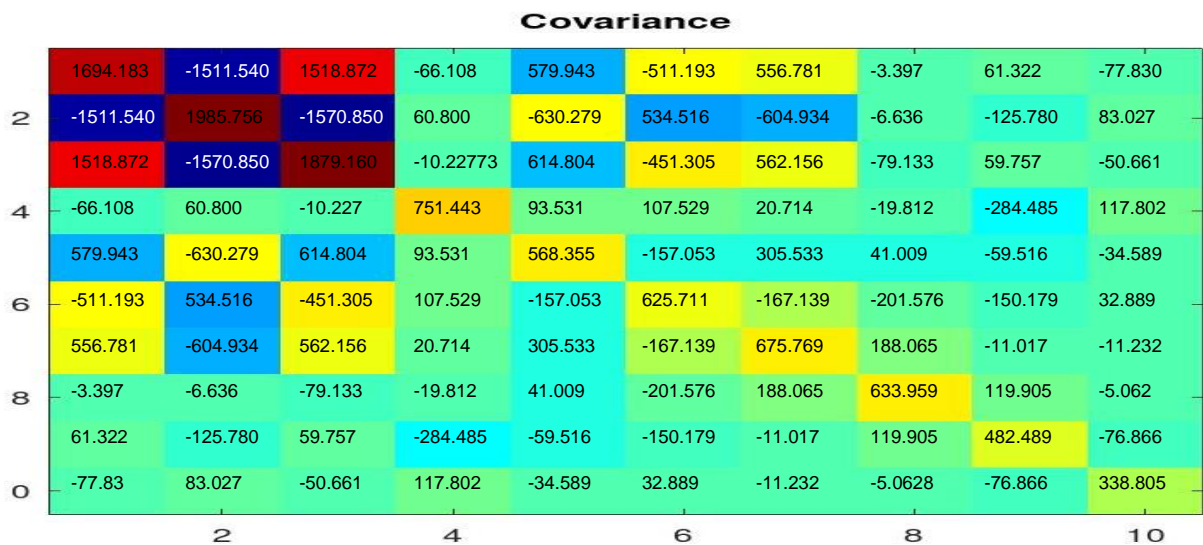


Figure 38 : BCM - Partial ridge regression  
**Partial ridge regression**

		-2.809	4.260	-0.544	0.894	-1.759	0.907	-0.307	-0.442	-0.309
2	-2.809		-2.933	-0.045	-1.331	1.257	-1.190	0.511	-0.942	0.205
	4.260	-2.933		0.358	1.189	-0.181	0.806	-1.435	0.385	0.308
4	-0.544	-0.045	0.358		1.016	0.487	-0.096	0.491	-3.077	1.286
	0.894	-1.331	1.189	1.016		0.098	1.212	0.636	-1.005	-0.544
6	-1.759	1.257	-0.181	0.487	0.098		0.728	-2.386	-1.104	-0.166
	0.907	-1.190	0.806	-0.096	1.212	0.728		2.754	-0.703	0.189
8	-0.307	0.511	-1.435	0.491	0.636	-2.386	2.754		1.528	-0.031
	-0.442	-0.942	0.385	-3.077	-1.005	-1.104	-0.703	1.528		-0.688
10	-0.309	0.205	0.308	1.286	-0.544	-0.166	0.189	-0.031	-0.688	
		2		4		6		8		10

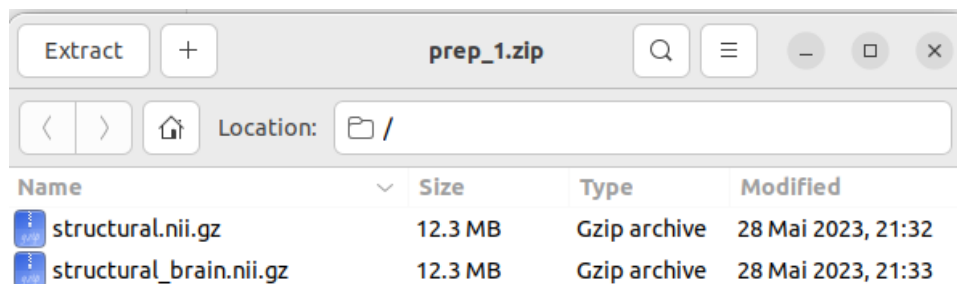
Even if the medical correctness of the data is not controlled, the fact that the same input file generated different output when changing just one parameter, implies that parameter 10 is working. The title of the BCM corresponds to the input parameter 12.

### Output: preprocessing

Each preprocessing process varies. So much so, that it would go beyond the scope of this thesis to check each parameter individually. Specially, because each parameter can generate additional output files and therefore, not only the correctness of the parameter's application on the output would need to be tested, but also if the generated files correspond to the expected number and type of files.

For illustration purpose only, a basic brain extraction will be run on a sMRI file. The output folder contains the unprocessed (input file) and the preprocessed sMRI file, as seen in Figure 39. In other preprocesses, many additional files can be generated.

Figure 39 : Preprocessing - brain extraction

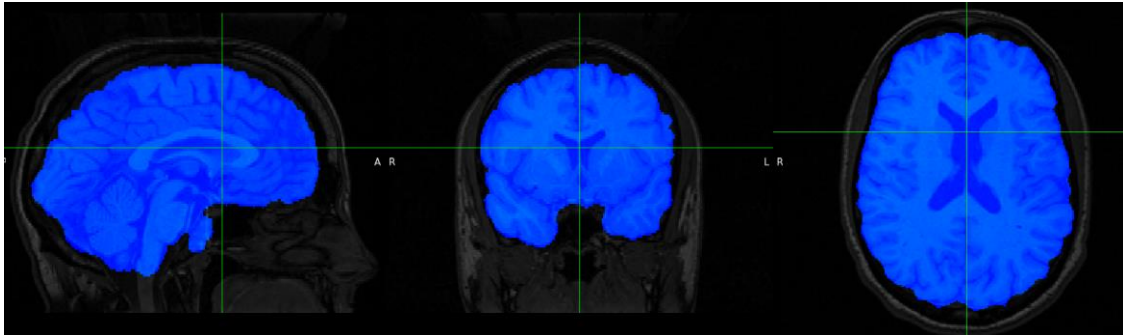


To check if the procedure worked, the FSLeves DIOCOM-viewer can be used. Figure 40 shows the unprocessed brain (white-and-black-coloured) which is overlaid by the



extracted brain which is blue-coloured. It can be concluded that the brain extraction was applied successfully.

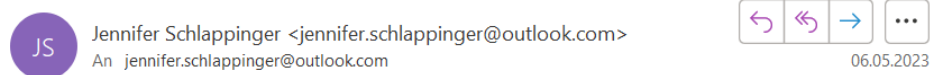
Figure 40 : BET extracted brain



### Output: reset password

Last and least, it will be checked if the user really receives a mail with a functioning link when requesting a password reset. As expected, the user, in this case the author, receives a mail with a link, that once clicked brings him back to reset password page.

Figure 41 : Mail when requesting a password reset



## You requested a new password.

Please click on the following link to reset : <http://localhost/resetPassword>

### 4.4.3 Test data

To test the application's functionalities two main sources were used:

**Source 1:** Flanker task (event related) retrieved from <https://legacy.openfmri.org/dataset/ds000102/>, containing 26 samples. Revision 2.0.0, - 27-May-2016 [Last accessed 06-May-2023]. The data can simply be downloaded as a zip file via the indicated link.

**Source 2:** FSL, Downloading and Installing FSL Course Data, retrieved from <https://open.win.ox.ac.uk/pages/fslcourse/website/downloads.html>. FSL explains in a step-by-step manual how the FSL Course Data can be downloaded. As indicated on their website, these sources are "provided for educational use only, not for research".

## 4.5 Installation and employment

### 4.5.1 Installation

This web application will only run in a Linux or Mac environment. If this condition is given, the installation guide's instructions can be followed. The user manual is also available in the GitHub repository directly.

- Link to the GitHub repository: <https://github.com/stefankam/predprodalzheimer/>
- Installation manual, Appendix 1

#### 4.5.1.1 Project structure

As mentioned in the requirement specifications, the application's extensibility is elementary. Hence, the project is clearly structured and components can easily be found, deleted and added without jeopardising the functioning of the entire project.

##### **fsl-backend;**

All the backend, and API-services will be found in this folder.

##### **api\_services;**

The API-services, connecting the frontend to the backend are managed here.

##### **database-services;**

There are two type of files in this folder. One is the service that manages the database connection (cnx\_db), while the others send and receive the database request per SQL-Table.

##### **helper\_services;**

All services that are particular to this project, such as the FSL and Octave interfaces are managed here.

##### **tests;**

all unit tests, can be developed and tested in this folder's files.

*settings.py* : This file will tell the backend if it should be run in production, testing or development mode. Note, that changes should be made in the .env/ .flaskenv file, if they are requirement dependant, such as personal passwords etc.

*.flaskenv* : Global setting file for Flask configurations

*requirements.txt* : Here is an exhaustive list of all npm requirements needed for this project.

##### **src;**

The entire frontend is implemented here.

##### **assets;**

All images and other are stored here.

##### **components;**

New components that can be used across the entire application such as Footers and Headers

##### **pages;**

The largest part of the frontend is built here

*routes.js* : This file handles navigation once the user is logged in.

*index.js* : From here the App is rendered. Change should be made with caution.

### 4.5.2 Employment

An audio-visual guide on using the application to its full extent can be found in the application itself.

## 5. Conclusion

The number of web applications is gradually increasing, and the healthcare industry is inclined to follow this evolution [47]. But it is still a long path to go. As for now, no web application on the market would suit the need for MRI processing and analysing tool. The reason for this shortage is namely due to security and privacy concerns of patients and medical professionals alike, but further investigations are needed to complete the list and find further reasons for the overall absence of medical web applications [41,48].

Next to instant accessibility and a more user-oriented interface as the major benefits of this FSL web application, the primary asset is the application's extensibility. With the rise of artificial intelligence and big data, MRI data could be faster and more precisely analysed and interpreted, thereby accelerating the predictions of diseases, notably Alzheimer, in this context. The application generated for this thesis is, therefore, a framework on which further functionalities can be added and an opportunity to automate the processing of medical (big) data. Other programs and tools that make use of AI algorithms can easily be added on top of the base frame of the FSL web application and multiple datasets of brain connectivity matrices can be compared and interpreted by the computer, potentially revealing a breakthrough in Alzheimer-linked research.

With the implementation of an FSL web application, a possible structure of what a medical web application could look like is created and the advantages of the different technologies in the context of a medical web application are underlined.

# Bibliography

- [1] K. Ganasegeran, A. Swee Hock Ch'ng, and I. Looi, "Handbook of Decision Support Systems for Neurological Disorders". *Elsevier Inc*, 2021, p 71 - 84.
- [2] WHO, "Dementia: Key Facts," *World Health Organization*, 20-Sep-2022. [Online]. Available at <https://www.who.int/en/news-room/fact-sheets/detail/dementia>. [Accessed: 06-May-2023].
- [3] National Institute on Aging, "What happens to the brain in alzheimer's disease?," *National Institute on Aging*, 08-Jul-2021. [Online]. Available: <https://www.nia.nih.gov/health/what-happens-brain-alzheimers-disease#:~:text=In%20Alzheimer's%20disease%2C%20as%20neurons,significant%20loss%20of%20brain%20volume>. [Accessed: 06-May-2023].
- [4] N. Amoroso, M. La Rocca, S. Bruno, T. Maggipinto, A. Monaco, R. Bellotti, & S. Tangaro, 2017. "Brain structural connectivity atrophy in Alzheimer's disease". *ArXiv*. /abs/1709.02369. [Online]. Available at <https://arxiv.org/pdf/1709.02369.pdf>. [Accessed: 06-May-2023].
- [5] M. Behroozi, M. Reza Daliri, "Software Tools for the Analysis of Functional Magnetic Resonance Imaging". *Basic and Clinical Neuroscience*, vol. 3, no. 5, p. 71 – 83, Autumn 2012. [Online]. Available at [https://applications.emro.who.int/imemrf/Basic\\_Clin\\_Neurosci/Basic\\_Clin\\_Neurosci\\_2012\\_3\\_5\\_71\\_83.pdf](https://applications.emro.who.int/imemrf/Basic_Clin_Neurosci/Basic_Clin_Neurosci_2012_3_5_71_83.pdf). [Accessed: 06-May-2023].
- [6] J. A. Mumford, T. E. Nichols, R. A. Poldrack. "Handbook of Functional MRI Data Analysis". *USA: Cambridge University Press*, 22-August-2011. ISBN:9781139498364.
- [7] B. Balusamy, N. A. R., and A. H. Gandomi, "Big Data: Concepts, technology and Architecture". *Hoboken NJ: John Wiley and Sons, Inc.*,13-April-2021. p,254.
- [8] T. Lowdermilk, "User-Centered Design: A Developer's Guide to Building User-Friendly Applications". *USA: O'Reilly Media*, 29-March-2013. ISBN: 9781449359836.
- [9] A. L. Weber, "History of head and Neck Radiology: Past, present, and future," *Radiology*, vol. 218, no. 1, pp. 1–4, Jan. 2001.
- [10] G. Xue, C. Chen, Z.-L. Lu, and Q. Dong, "Brain imaging techniques and their applications in decision-making research," *Xin li xue bao. Acta psychologica Sinica*, 03-Feb-2010. [Online]. Available at <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2849100/>. [Accessed: 06-May-2023].
- [11] A. Berger, "How does it work?: Magnetic resonance imaging," *BMJ*, vol. 324, no. 7328, pp. 35–35, Jan. 2002.
- [12] T. Wager, "Principles of fMRI Part 1, Module 2: Analysis of fMRI Data." *YouTube*, 2015. [Online]. Available at <https://www.youtube.com/watch?v=tQyMqRqHwao&list=PLfXA4opIOVrGHncHRxI3Qa5GeCSudwmxM&index=2>. [Accessed: 06-May-2023].
- [13] FSL, "FSL Course ". *open.win.ox.ac.uk*; Beijing, 2019. [Online]. Available at [https://open.win.ox.ac.uk/pages/fslcourse/website/online\\_materials.html](https://open.win.ox.ac.uk/pages/fslcourse/website/online_materials.html). [Accessed: 06-May-2023].
- [14] D. A. Micheau and D. D. Hoa, "Brain MRI 3D: Normal anatomy: E-anatomy," *IMAIOS*, 05-Oct-2022. [Online]. Available: <https://www.imaios.com/en/e-anatomy/brain/mri-brain>. [Accessed: 07-May-2023].
- [15] The University of Edinburgh, "Functional Mr (fmri)," *The University of Edinburgh*, 23-Jun-2021. [Online]. Available at <https://www.ed.ac.uk/clinical-sciences/edinburgh>.

imaging/research/themes-and-topics/medical-physics/imaging-techniques/functional-mr-fmri. [Accessed: 06-May-2023].

- [16] K. Chee Keong, "Vulnerability to sleep deprivation: A Drift Diffusion Model Perspective," *Neurobit Inc.*, pp. 86–88, Apr. 2015.
- [17] I. Cifre, M. Zarepour, S. G. Horovitz, S. A. Cannas, and D. R. Chialvo, "Further results on why a point process is effective for estimating correlation between brain regions," *Papers in Physics*, vol. 12, pp. 120003–2-120003–4, Jun. 2020.
- [18] L. M. Hocke, Y. Tong, and B. de B. Frederick, "An automatic motion-based Artifact Reduction Algorithm for fNIRS in concurrent functional magnetic resonance imaging studies (amara-fmri)," *Algorithms*, vol. 16, no. 5, p. 230, Apr. 2023.
- [19] W. S. Sohn, T. Y. Lee, K. Yoo, M. Kim, J.-Y. Yun, J.-W. Hur, Y. B. Yoon, S. W. Seo, D. L. Na, Y. Jeong, and J. S. Kwon, "Node identification using inter-regional correlation analysis for mapping detailed connections in resting state networks," *Frontiers in Neuroscience*, vol. 11, no. TECHNOLOGY REPORT article, May 2017.
- [20] A. J. C. Eijlers, M. M. Schoonheim, J. J. G. Geurts, L. Douw, K. A. Meijer, and A. M. Wink, "Functional network dynamics on functional MRI: A Primer on an emerging frontier in neuroscience," *Radiology*, Jun-2019. [Online]. Available at <https://pubmed.ncbi.nlm.nih.gov/31237814/>. [Accessed: 06-May-2023].
- [21] Y. Diao, T. Yin, R. Gruetter, and I. O. Jelescu, "Piracy: An optimized pipeline for functional connectivity analysis in The rat brain," *Frontiers in Neuroscience*, vol. 15, Mar. 2021.
- [22] Law insider, "Medical applications definition," Law Insider. [Online]. Available: <https://www.lawinsider.com/dictionary/medical-applications#:~:text=Medical%20Applications%20means%20diagnostic%20products,those%20uses%2C%20including%20without%20limitation%2C>. [Accessed: 06-May-2023].
- [23] Technostacks, "Different types of application software: Technostacks," *Technostacks Infotech*, 13-Jan-2023. [Online]. Available at <https://technostacks.com/blog/types-of-application-software/>. [Accessed: 06-May-2023].
- [24] Simplilearn, "What is application software? (with examples): Simplilearn," *Simplilearn.com*, 25-Apr-2023. [Online]. Available at <https://www.simplilearn.com/tutorials/programming-tutorial/what-is-application-software>. [Accessed: 07-May-2023]
- [25] T. Vinay. "How to Speak Tech: The Non-Techie's Guide to Key Technology Concepts." Germany: *Apress*, 26-March-2019, p. 109 – 120
- [26] R. Ye, ".NET MAUI Cross-Platform Application Development". *Packt Publishing*, 2023, vol 1, p. 4-7. ISBN: 978-1-80056-922-5
- [27] Vinugayathri, S. Dighe, and A. Deshpande, "5 javascript alternatives for Front End Development," *Build Offshore Technology Team in India. In No Time*, 2020. [Online]. Available at <https://www.clariontech.com/blog/5-javascript-alternatives-for-front-end-development>. [Accessed: 06-May-2023].
- [28] StackPath, "What is a web application?," *stackpath.com*. [Online]. Available at <https://www.stackpath.com/edge-academy/what-is-a-web-application/>. [Accessed: 06-May-2023].
- [29] HooSuite and we are social, "Digital 2022: Global Overview Report - DataReportal – Global Digital Insights," *DataReportal*, 04-May-2022. [Online]. Available: <https://datareportal.com/reports/digital-2022-global-overview-report>. [Accessed at 06-May-2023].

- [30] J. Desai, "Web application vs Desktop Application: Pros and cons," *POSITIWISE*, 15-Apr-2022. [Online]. Available at <https://positiwise.com/blog/web-application-vs-desktop-application-pros-and-cons/>. [Accessed: 06-May-2023].
- [31] S. L. Fowler and V. R. Stanwick, *Web application design handbook: Best practices for web-based software*. Amsterdam: *Elsevier*, 2004, Ch. 1. ISBN 9780080481708.
- [32] G. Nizamettin, and K. Nitin. "Building Hybrid Android Apps with Java and JavaScript: Applying Native Device APIs". USA: *O'Reilly Media*, 23-July-2013. ISBN: 9781449361877.
- [33] AFNI Installation
- [34] AFNI, Documentation, Version 32.1.04. *National Institute of Mental Health*, Last compile 05-May-2023. Available at [https://afni.nimh.nih.gov/pub/dist/doc/html/doc/background\\_install/install\\_instructs/steps\\_windows10.html](https://afni.nimh.nih.gov/pub/dist/doc/html/doc/background_install/install_instructs/steps_windows10.html). [Accessed: 06-May-2023].
- [35] CONTRIBUTORS (<https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Contributors>), FMRIB software library, Version 6.0, *FSL*, 20-Feb-2023. [Online]. Available at <https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FSL>. [Accessed: 06-May-2023].
- [36] R. Goebel, "Products - brainvoyager," *Brain Innovation*, May-2015. [Online]. Available at <https://www.brainvoyager.com/products/brainvoyager.html>. [Accessed: 07-May-2023].
- [37] Wikipedia. "Open source". *Wikipedia*, 2023. [Online]. Available at [https://en.wikipedia.org/wiki/Open\\_source](https://en.wikipedia.org/wiki/Open_source)
- [38] T. Petroc , "Operating systems market share of desktop PCs 2013-2023, by month." 27-Feb-2023. [Online]. Available at <https://www.statista.com/statistics/218089/global-market-share-of-windows-7/#:~:text=Microsoft%20Windows%20was%20the%20dominant,in%20the%20desktop%20OS%20market>. [Accessed: 06-May-2023].
- [39] A. Khrua, "Top Healthcare Apps," *QUARA*, 24-Mar-2023. [Online]. Available at <https://qarea.com/blog/top-healthcare-apps>. [Accessed: 06-May-2023].
- [40] Virtual Group Expo, "Medical Technical Facilities," *The B2B marketplace for medical equipment*. [Online]. Available at <https://www.medicalexpo.com/medical-manufacturer/medical-web-application-11312.html>. [Accessed: 06-May-2023].
- [41] K. Fultz Hollis, "To share or not to share: What motivates researchers to share their data?," *AMIA Jt Summits Transl Sci Proc.*, pp. 420–427, Jul. 2016. [Online]. Available at <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5001759/>. [Accessed: 06-May-2023].
- [42] FMRIB Software Library. FMRIB Software Library Manuals, version 3.2 – September 2004. [Online]. Available at <https://poc.vl-e.nl/distribution/manual/fsl-3.2/>. [Accessed: 06-May-2023]
- [43] Barkhof, S. Haller, and S. A. Rombouts, "Resting-state functional Mr Imaging: A new window to the brain," *Radiology*, vol. 272, no. 1, pp. 1–49, Apr. 2014.
- [44] A. Blazejewska, "Introduction to MRI data processing with FSL", *Why N How Seminar, Martinos Center*, 30-March-2017. [Online]. Available at [https://gate.nmr.mgh.harvard.edu/wiki/whynhow/images/8/8f/WhyNhow\\_FSL\\_final-WEB.pdf](https://gate.nmr.mgh.harvard.edu/wiki/whynhow/images/8/8f/WhyNhow_FSL_final-WEB.pdf). [Accessed: 06-May-2023].
- [45] A. Jahn, "Andy's Brain Book," *University of Michigan Sp.*, 2019. [Online]. Available at <https://andysbrainbook.readthedocs.io/en/latest/>. [Accessed: 06-May-2023]
- [46] Chou and al. *AJNAR*, "rPython/FSL Resting State Pipeline," *Brain Imaging & Analysis Cente*, May-2012. [Online]. Available: [https://wiki.biac.duke.edu/biac:analysis:resting\\_pipeline](https://wiki.biac.duke.edu/biac:analysis:resting_pipeline). [Accessed: 06-May-2023].

- [47] H. Mukhtar, H. F. Ahmad, M. Z. Khan, and N. Ullah, "Analysis and evaluation of COVID-19 web applications for Health Professionals: Challenges and opportunities," *Healthcare (Basel, Switzerland)*, 07-Nov-2020. [Online]. Available at <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7712438/>. [Accessed: 06-May-2023].
- [48] C. Mata Miquel, "Web-based application for Medical Imaging Management," *Universitat de Girona*, 2015. [Online]. Available at <https://dugi-doc.udg.edu/bitstream/handle/10256/11660/tcmm1de1.pdf?sequence=5>. [Accessed: 06-May-2023]
- [49] Airfocus. *MoSCoW prioritization*. Airfocus, 24-July-2020. [Online]. Available: <https://airfocus.com/glossary/what-is-moscow-prioritization/#:~:text=%E2%80%9CMoSCoW%E2%80%9D%20is%20an%20acronym%20for,included%20to%20help%20with%20pronunciation>. [Accessed: 06-May-2023].
- [50] Mayo Clinic, "EEG (electroencephalogram)," Mayo Clinic, 11-May-2022. [Online]. Available: <https://www.mayoclinic.org/tests-procedures/eeg/about/pac-20393875>. [Accessed: 06-May-2023].
- [51] D. Larimer, "Blockchain is a better application server and database," *More Equal Animals*, 10-Feb-2020. [Online]. Available at <https://moreequalanimals.com/posts/blockchain-is-better-application-server-and-database>. [Accessed: 06-May-2023].
- [52] D. Ruibo, "A web application's limited resources - web application performance and scalability", *Beginning Django*, 2015. [Online]. Available at <https://www.webforefront.com/performance/limitedresources.html>. [Accessed: 06-May-2023].
- [53] J. Brown, "5 SMART goals for a QA analyst". Boston University, 26-August-2020. Available at <https://www.techtarget.com/searchsoftwarequality/feature/Goal-1-for-the-QA-tester-Take-ownership>. [Accessed: 06-May-2023]
- [54] N. Kostyshak, "Healthcare Web Application Development: An ultimate guide," *OTAKOYI*, 03-Nov-2022. [Online]. Available at <https://otakoyi.software/blog/healthcare-web-application-development-an-ultimate-guide>. [Accessed: 06-May-2023].
- [55] ResearchComputing, "Documentation/coding-best-practices.md at main · researchcomputing/documentation," *GitHub*, 06-May-2022. [Online]. Available at <https://github.com/ResearchComputing/Documentation/blob/main/docs/programming/coding-best-practices.md>. [Accessed: 06-May-2023].
- [56] Meta Open Source. "React – The library for web and native user interfaces," *React*, 2023. [Online]. Available at <https://react.dev/>. [Accessed: 06-May-2023].
- [57] M. Lutz, *Python: Kurz Gut*, vol. 5. Germany Köln: O'Reilly Verlag, 2014.
- [58] Python Software Foundation. Python Language Reference, version 2.7. Available at <http://www.python.org>
- [59] Wikipedia. "Python (Programmiersprache)," *Wikipedia*, 04-May-2023. [Online]. Available at [https://de.wikipedia.org/wiki/Python\\_\(Programmiersprache\)](https://de.wikipedia.org/wiki/Python_(Programmiersprache)). [Accessed: 06-May-2023].
- [60] Oct2Py Contributors, "Oct2Py: Python to GNU octave bridge," *Oct2Py*. Version 4.0.6. [Online]. Available: <https://blink1073.github.io/oct2py/>. [Accessed: 06-May-2023].
- [61] PostgreSQL, Ed, "Documentation," *PostgreSQL*. [Online]. Available at <https://www.postgresql.org/docs/current/>. [Accessed: 07-May-2023].
- [62] Google Cloud, Ed., "PostgreSQL im Vergleich zu SQL server: Was ist der unterschied?," *Google*. [Online]. Available at <https://cloud.google.com/learn/postgresql-vs-sql?hl=de>. [Accessed: 06-May-2023].

- [63] R. Sears, C. van Ingen, and J. Gray, "To BLOB or Not To BLOB: Large Object Storage in a Database or a Filesystem ». *Microsoft Research Microsoft Corporation One Microsoft Way*, Redmond, WA, tech., 2006. Available at <https://www.microsoft.com/en-us/research/publication/to-blob-or-not-to-blob-large-object-storage-in-a-database-or-a-filesystem/?from=https://research.microsoft.com/apps/pubs/default.aspx?id=64525&type=exact>. [Accessed: 06-May-2023]
- [64] GNU Octave. GNU Octave Documentation, version 8.2.0. Available at <https://docs.octave.org/v8.2.0/Displaying-Images.html>.
- [65] J. B. Poline and M. Brett, "Sometimes, the NIFTI image stores the TR in the header," *Psych 214 – functional MRI methods*, 2016. [Online]. Available at [https://bic-berkeley.github.io/psych-214-fall-2016/tr\\_and\\_headers.html](https://bic-berkeley.github.io/psych-214-fall-2016/tr_and_headers.html). [Accessed: 06-May-2023].
- [66] Patrick, "What is SHA-256 and how is it related to bitcoin?," Mycryptopedia, 26-Apr-2022. [Online]. Available: <https://www.mycryptopedia.com/sha-256-related-bitcoin/>. [Accessed: 06-May-2023].
- [67] "How to build an OAuth service using Python, Flask, postgres and JWT," *Grizzly Peak Software*, Feb-2020. [Online]. Available at <https://www.grizzlypeaksoftware.com/articles?id=5SCpQMgookgKNtupzNHg9K>. [Accessed: 06-May-2023].
- [68] Nypipe contributors (<https://github.com/nipy/nipype/graphs/contributors>), "Neuroimaging in Python - Pipelines and Interfaces - nipype pipeline and interfaces package", Release 1.8.7. [Online]. Available at <https://nipype.readthedocs.io/en/latest/api/generated/nipype.interfaces.fsl.html>. [Accessed: 06-May-2023].
- [69] A. Norden, "Learn software testing in 24 Hours," *Google Books*, 31-Oct-2020. [Online].
- [70] S. Felice, "Top testing libraries for react in 2023," *BrowserStack*, 14-Feb-2023. [Online]. Available: <https://www.browserstack.com/guide/top-react-testing-libraries>. [Accessed: 06-May-2023]
- [71] K. Pykes, "How to Use Pytest for Unit Testing". *Datacamp*, July-2022. Available at <https://www.datacamp.com/tutorial/pytest-tutorial-a-hands-on-guide-to-unit-testing>. [Accessed: 06-May-2023]
- [72] J. Ashburner, K.J. Friston, 1997. *The role of registration and spatial normalization in detecting activations in functional imaging*. *Clinical MRI/Developments in MR*, 7(1):26-28.
- [73] W. C. Robert, 1996. *AFNI: software for analysis and visualization of Functional Magnetic Resonance Neuroimages*. *Computer and Biomedical Research*, 29:162-173.
- [74] S. M. Smith, M. Jenkinson, M. W. Woolrich, C. Beckmann, F. and et al., 2004. *Advances in functional and structural MR image analysis and implementation as FSL*. *Neuroimage*, 23: S208-S219.
- [75] R. Goebel, F. Esposito, E. Formisano, 2006. *Analysis of FIAC data with BrainVoyager QX: From single-subject to cortically aligned group GLM analysis and self-organizing group ICA*. *Human Brain Mapping*. 27(5): 392-401.
- [X] M. L. Stanley, M. N. Moussa, B. M. Paolini, R. G. Lyday, J. H. Burdette, and P. J. Laurienti, "Defining nodes in complex brain networks," *Frontiers in Computational Neuroscience*, vol. 7, Nov. 2013.
- [X] M. Saeidi, W. Karwowski, F. V. Farahani, K. Fiok, P. A. Hancock, B. D. Sawyer, L. Christov-Moore, and P. K. Douglas, "Decoding task-based fmri data with graph neural networks, considering individual differences," *Brain Sciences*, vol. 12, no. 8, p. 1094, 2022.



- [X] Nijholt Anton, S. Nam Chang, Lotte Fabien, 2018. Brain–Computer Interfaces Handbook: Technological and Theoretical Advances. USA: CRC Press, 9. Januar 2018. ISBN 9781351231947
- [X] D. Ahmedt-Aristizabal, M. A. Armin, S. Denman, C. Fookes, and L. Petersson, “Graph-based deep learning for medical diagnosis and analysis: Past, present and future,” *Sensors*, vol. 21, no. 14, p. 4758, Jul. 2021.
- [X] Vemuri, P., Jack, C.R, 2010. Role of structural MRI in Alzheimer's disease. *Alz Res Therapy* 2, 31 August 2010. <https://doi.org/10.1186/alzrt47>
- [X] BrainFacts/SfN, “The neuron,” BrainFacts.org, 01-Apr-2012. [Online]. Available at <https://www.brainfacts.org/brain-anatomy-and-function/anatomy/2012/the-neuron>. [Accessed: 06-May-2023].

# Appendix 1: Installation manual for developers

The FSL-web-application visualises and simplifies the use of FSL. The goal of this document is, hence, to help developers to quick and easily get started with the setup of this project.

Please note, since FSL is only runnable on Linux and MacOs, this guide does not provide any information on specific windows application options. If you are a windows user, please use a virtual machine for this set up procedure.

## 1: Downloading the project

- 1) Go to GitHub and download the project via the following link:  
<https://github.com/stefankam/predprodalzheimer>, Alternatively, you can clone the files directly into you own local directory.

## 2: Installing FSL and Octave (you could also use MATHLAB)

- 1) To install FSL, follow the instructions given on the official FSL website, <https://fsl.fmrib.ox.ac.uk/fsl/fslwiki/FslInstallation>. Once installed, simply type *fsl* in the terminal to check if the application is running. Make sure, that FSL is saved to your computer's environment. (\$FSLDIR).
- 2) Install octave. This step should be runnable without any further issues

## 3: Checking the pre-requirements

- 1) Firstly, you will have to make sure you will have to make sure that Python is available on your computer.
- 2) Also npm has to be downloaded and accessible on your computer. To do so navigate to the following link to access the nodejs packages  
<https://nodejs.org/en/download/>
- 3) Then verify if the package PIP ("PIP Installs Packages") is installed and up to date. You can do that by following the instructions below:

```
> python3 -m pip install --user --upgrade pip  
> python3 -m pip --version  
> pip --version
```

If the outcome of this last command is somewhat like:

```
pip 22.0.2 from C:\Users\...\Programs\Python\Python310\lib\site-packages\pip  
(python 3.10)., pip was successfully installed.
```

- 4) Finally, check if you have the virtual environment package for python, and if not, install it.

```
> python3 -m pip install --user virtualenv
```

Once all pre-requirements met, you can move on to point four.

#### 4: Setting up your virtual environment

Open the downloaded project in a code editor of your preference and via the terminal navigate to: `cd ../predprodalzheimer-scratch/fsl_frontend/fsl_backend`

- 1) And there create your virtual environment:

```
> python3 -m venv [yourVirtualEnvironmentName]
```

If your virtual environment was created, a new folder carrying the name of [yourVirtualEnvironmentName] will appear.

Now, you should be able to activate and deactivate your virtual environment by typing the following command in your terminal:

```
> source env/bin/activate
```

Once activated the name of your virtual environment should appear in parentheses.

To deactivate the virtual environment simply type *deactivate*.

#### 5: Installing packages

With the activated virtual environment, you can proceed with the installation on the various packages needed.

- 1) Therefore simply run :

```
> python3 -m pip install -r requirements.txt
```

which will install all the packages listed in the requirements.txt.

- 2) Next, installing npm, but make sure your virtual environment is not activated.

```
> sudo apt install npm
```

A folder containing the relevant node modules should appear.

- 3) Finally you can configure the flaskenv and .env files, so you can access the PostgreSQL database of your choice.

- 4) Run the database script, that is located in the project's folder postgresql.