

# GaitQualityAnalyzer Documentation

version1

## 1 Installation and Data Import

The system GaitQualityAnalyzer is wrapped in the form of Docker containers to be executable on different platforms.

### 1.1 Hardware requirements

The recommended requirements on hardware are 16 GB RAM and 20 GB disk space.

### 1.2 How to initially build the system

1. Install docker and docker-compose on your local machine.
2. Download the sources of GaitQualityAnalyzer, open a terminal and navigate to the directory with the downloaded sources.
3. Generate the database tables by editing the file `/src/main/resources/application-prod.properties` and changing the following line:

```
spring.jpa.hibernate.ddl-auto=create
```

4. Run the following command to build and start the GaitQualityAnalyzer:

```
docker-compose --profile prod up --build
```

This process will start the following services:

- *Database* available at <http://localhost:5432>.
- *Website* accessible at <http://localhost:8080>.
- *Gait cycle quality extractor* available at <http://localhost:65434>.
- *Gait cycle feature vector extractor* available at <http://localhost:65432>.

### 1.3 How to run the system

If the system is built using the process specified in the previous steps, it is automatically running at: `http://localhost:8080`. Otherwise, check that the file `/src/main/resources/application-prod.properties` has the following line set to `validate`:

```
spring.jpa.hibernate.ddl-auto=validate
```

then run the following command to start the GaitQualityAnalyzer:

```
docker-compose --profile prod up
```

In case there is some change in the configuration or source files, there is a need to rebuild the system by executing the following command:

```
docker-compose --profile prod up --build
```

### 1.4 How to stop the system

If you want to properly shut down the system you can either cause interrupt in the terminal or from the root directory run the following command:

```
docker-compose down
```

### 1.5 How to backup the system

GaitQualityAnalyzer automatically backups the database each day. The backup files are located in the `/database/backup` directory. The directory contains a structure of daily, weekly, and monthly backups. Each folder contains a zip of sql files that were exported using `pg_dump`.

Due to safety reasons, it is recommended to back up the latest exported zip files to an external location (by simple file-copy approach).

### 1.6 How to populate GaitQualityAnalyzer with data

To enable the data population to run on start of the application, edit the file `/src/main/resources/application-prod.properties` and change the following line to:

```
application.data-population.enabled=true
```

The property, `application.data-population.enabled`, tells the system to run the data population scripts when the system starts up. This will insert the data from the `/src/main/resources/animations` folder into the database. If you want to reset the database before the start of the application, change the following line in the same file to:

```
spring.jpa.hibernate.ddl-auto=create
```

### 1.6.1 File format

Data are uploaded in the form of normalized gait cycles as the output of an external data pre-processor – this pre-processor accompanying GaitQualityAnalyzer is applied to the motion data recorded by the Vicon motion capture technology. Specifically, each gait cycle consists of kinematic/kinetic characteristics of both left and right lower limbs that are stored in two separate CSV files. Each file must be named in the following format:

```
<Lastname><Firstname><BirthYear:YY>–  
<VisitDate:DDMMYYYY>–Cal–<VisitId>–(L|R).csv
```

The CSV files contain recordings of the patient’s gait characterized by various attributes such as 3D coordinates of markers, joint-angle rotations, or joint-moment power, denoted like: *LFEO\_x*, *LFEO\_y*, *LFEO\_z*, *LFEA\_x*, ..., *LHipAngles\_x*, *LHipAngles\_y*, ..., *LGroundReactionForce\_x*, ..., *RFEO\_x*, ..., *RHipAngles\_x*, ...

Such file has a two-row header, where the second row must contain the attribute with the prefixed side. Each attribute is represented by three columns *\_x*, *\_y* and *\_z*. For example, the right side (*-R.csv*) must contain the headers *RPelvisAngles\_x*, *RPelvisAngles\_y*, etc. Additionally, both sides should contain side-less attributes (without the L or R prefix): *PELO*, *PELA*, *PELL*, *PELP*

The values in each column must equal to 101, which corresponds to the standard length of the normalized gait cycle.

### 1.6.2 Population folders

To populate the system with data, upload the CSV files into the */src/main/resources/animations* folder. When the *application.data-population.enabled* is set to *true*, the system automatically creates a patient, visit, and corresponding gait cycle based on the structured information encoded directly in the filename. The patient’s data which are not included in the filename, such as gender or e-mail, are set to default/random values. The gait cycle data are associated with a visit only in case when both CSV files corresponding to left and right limbs are provided.

To assess the quality of human gait, a default implementation of feature quality extractor is provided. This implementation computes the difference in values of pre-defined attributes of patients’ gait cycles with respect to the *norm* gait cycles. This requires uploading gait cycles that characterize the gait norm, into */src/main/resources/norm\_animations*, before running the extractor. The files to be uploaded have the same format as the files in the */src/main/resources/animations* folder.

## 1.7 How to use custom extractors

If you want to replace the integrated **gait cycle quality extractor** (see Section 5.1.1 for more details) or the **gait cycle feature vector extractor** (see

Section 5.2.1), you can do it by placing the new source code into *src/main/classifiers*. Then, in the *docker-compose.yml* file you modify the *feature\_quality* and *feature\_vector* services to point to the new extractors. In the system configuration */src/main/resources/application-prod.properties* file you change the following lines to:

```
# using domain or ip address
socket.server.deepFeatures.host=example.com
socket.server.deepFeatures.port=65432

# using name of the docker container instead of web address
socket.server.quality.host=feature_quality
socket.server.quality.port=65434
```

## 1.8 System architecture

The architecture of the system is composed of the following services.

- **web** – builds a Maven project which consists of backend and frontend:
  - **backend** – a Spring boot app. build with Java 11 and Tomcat 9;
  - **frontend** – a React application build with Typescript and Bootstrap.
- **database** – runs a PostgreSQL 13.4 database with the database schema depicted in Appendix in Figure 16.
- **feature\_quality** – runs a python 3.10 socket server with numpy and pandas.
- **feature\_vector** – runs a python 3.10 socket server with nympy, pandas, and pyTorch.

## 2 Introduction to GaitQualityAnalyzer

GaitQualityAnalyzer is a software system for analyzing the quality of human gait. The system manages human subjects along with their gait style recorded in different time periods in the form of motion capture data. The primary purpose is to provide functionality for searching for similar movement patterns on the level of gait cycles and assessing the quality of the retrieved patterns. This allows users to determine whether a subject's gait has improved or worsened after a particular event or treatment. Although GaitQualityAnalyzer was specifically designed to evaluate the effectiveness of treatments for patients with cerebral palsy, it can be potentially applied to other scenarios where the analysis of movement patterns is necessary.

The system is designed as a client-server architecture with a web-based graphical user interface. This interface allows users mainly to:

- Manage subjects, visits, and gait cycles associated with visits;
- Assess the quality of gait cycles;
- Search for similar visits based on attribute filters (such as age, tags, or visit date) and similarities of their associated gait recordings with respect to a selected query visit from another subject;
- Determine whether the most relevant retrieved subjects exhibit an improvement or deterioration of gait style in time.

In the following text, there is a more-detailed description of the main functionality of the provided graphical user interface.

### 3 Authentication

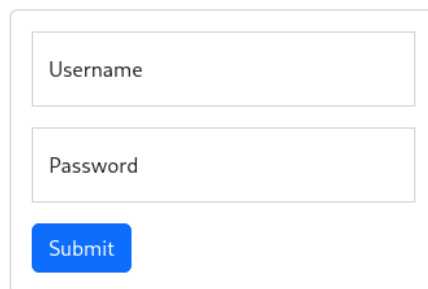
To sign in to the system, the administrator must create an account for the user first. Users who enter any web page will need to sign in.

#### 3.1 How to sign in

To sign in to the system, visit the home page <http://localhost:8080>. If you are not already signed in, the system will prompt you to enter your username and password (Figure 1).

The system has two privilege roles: USER and ADMIN. Compared to the USER role, the ADMIN role can additionally manage user accounts. The initial data population creates two default users – (1) the account with the USER role has username “demo” and password “pass”, (2) the account with the ADMIN role has username “admin” and password “pass”.

Once you are signed in, you will be redirected to a main dashboard where you can observe statistics of an underlying database with stored gait-cycle data.



The image shows a sign-in form with two input fields and a submit button. The first field is labeled 'Username' and the second is labeled 'Password'. Below the fields is a blue button labeled 'Submit'.

Figure 1: Sign in page

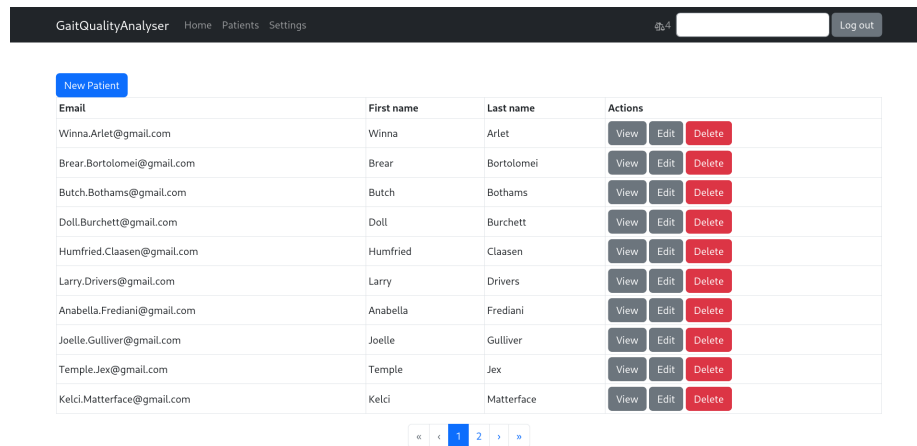
## 4 Patient

### 4.1 Manage a patient

In the navigation bar click on *Patients* (Figure 2). In the patient list page, you can add a new patient by pressing the *New patient* button. You will be prompted to enter their: email, first name, last name, social id, gender, date of birth, and optionally tags associated with that patient. The tags can be used for filtering patients based on the specified tags during the process of similarity searching.

### 4.2 List all patients

In the navigation bar click on *Patients* (Figure 2). This loads all the database patients and displays only 10 patients per page. You can see more patients by selecting the next page at the bottom. The number of patients per page can be configured in the `src/main/resources/application.properties` file by setting the `spring.data.web.pageable.default-page-size`.



Email	First name	Last name	Actions
Winna.Ariet@gmail.com	Winna	Ariet	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Brear.Bortolomei@gmail.com	Brear	Bortolomei	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Butch.Bothams@gmail.com	Butch	Bothams	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Doll.Burchett@gmail.com	Doll	Burchett	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Humfried.Claasen@gmail.com	Humfried	Claasen	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Larry.Drivers@gmail.com	Larry	Drivers	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Anabella.Frediani@gmail.com	Anabella	Frediani	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Joelle.Gulliver@gmail.com	Joelle	Gulliver	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Temple.Jex@gmail.com	Temple	Jex	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>
Kelci.Matterface@gmail.com	Kelci	Matterface	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a>

Figure 2: List of patients page

### 4.3 Search for a patient

In the navigation bar, the search bar allows users to search patients by their full name and the results are auto-completed based on partial matches, as can be seen in Figure 3; the interactive menu displays the matching patients.

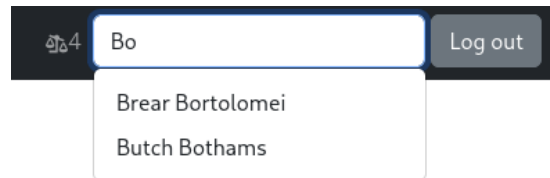


Figure 3: Patient search bar

#### 4.4 View patient details

From the list of patients page, click the *View* button or directly click on the patient's name to view the patient details. Besides the basic patient's information, the page displays a list of visits together with the recorded gait styles in the form of motion capture data (Figure 4).

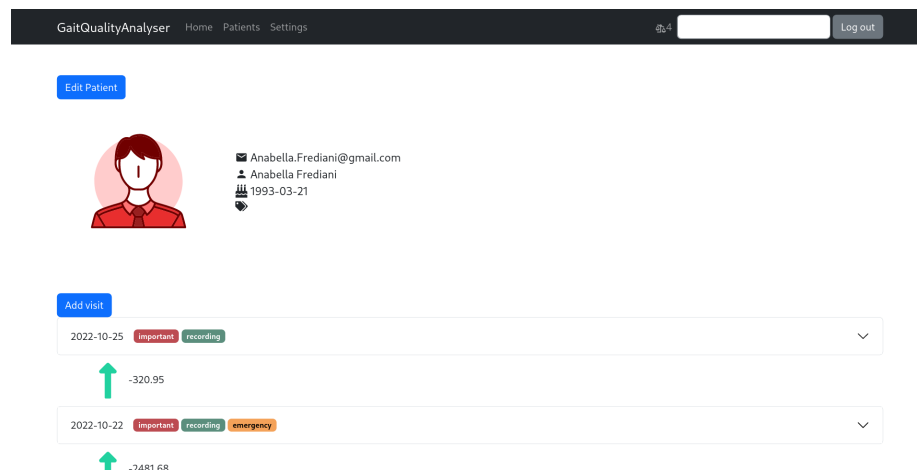


Figure 4: Patient page

### 5 Visit

A visit is an instance of the patient visiting a doctor. Each visit can have several gait cycles associated in the form of recorded motion capture data. The patient's gait is assumed to be recorded during their visit using a Vicon motion capture technology<sup>1</sup> and uploaded to the system in the format specified in Section 1.6.1 (the format is defined by a supplied external data pre-processor). The basic

<sup>1</sup><https://www.vicon.com/>

block is a *gait cycle* that corresponds to two synchronized footsteps – footstep of the left lower limb and footstep of the right lower limb. Each gait cycle is normalized in the temporal dimension and has exactly 101 frames.

On the patient’s detail page, click on the visit date to expand the visit details (Figure 5). This page allows users to manage basic visit information, manage visit’s gait cycles, find similar visits, and compare visits based on time series of user-defined attributes.

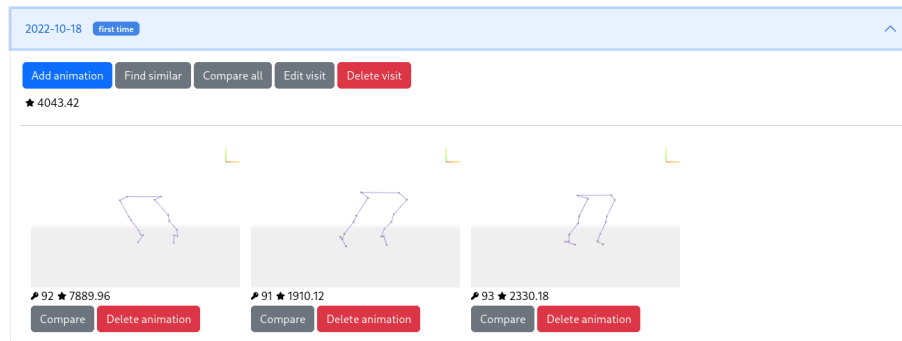


Figure 5: Visit details page

## 5.1 Quality of gait cycles and of the whole visit

Once you open the visit, there is a star icon with a real number representing the assessed quality of person’s gait associated with the visit. This value is the average of the qualities of gait cycles associated with the given visit. The higher the quality, the better the gait is. Between two consecutive patient’s visits, there is the arrow representing whether the gait quality has improved or worsened (Figure 4). If the arrow is *red*, the quality of the latter visit is *worse* then of the former. If the arrow is *green*, the quality of the latter visit is *better* then of the former.

### 5.1.1 Gait quality extractor

Gait quality extractor is a socket server, that takes a gait cycle as an input and returns its assessed quality in the form of a real number.

The default implementation of the service takes the input gait cycle and compares it with a list of norm gait cycles using the Euclidean distance based on a set of time series corresponding to predefined attributes. The minimum distance constitutes the dissimilarity to a given norm patient. This distance is normalized (by a pre-defined maximum distance) and subtracted from value 1. The higher the value, the higher gait quality is.



The default gait quality extractor can be replaced by any other implementation as specified in the installation notes. There is an external supplied implementation based on the Gait Deviation Index (GDI)<sup>2</sup>.

### 5.1.2 Extract qualities of all gait cycles

In the navigation bar, click on *Settings*. In the first table, find the row with *Extract quality* and press the *Extract* button (Figure 6). The system then loads all the gait cycles in the system and sends them to the Gait quality extractor (running at a specified port) to re-assess the quality of all stored gait cycles. This can be especially used when new visits/gait cycles are added or a new gait quality extractor is implemented.

Title	Last update	Action
Calculate deep features	10/22/2022 10:10:52 PM	<a href="#">Calculate</a>
Calculate quality	10/22/2022 10:11:11 PM	<a href="#">Calculate</a>

Figure 6: Re-calculation of the quality and re-extraction of deep features for all gait cycles stored in the database

## 5.2 Deep feature representation of gait cycles

Deep features constitute a fixed-length vector (e.g., 1,024 dimensions) of real numbers that reflect content-preserving characteristics of a recorded gait cycle. The vector is extracted by a deep feature extractor for each gait cycle. The deep features of two gait cycles (usually of two different patients) are compared by the Euclidean distance function which quantifies their dissimilarity. This function is employed during the process of similarity searching for identifying visits of patients with similar movement characteristics.

### 5.2.1 Deep feature extractor

The deep feature extractor is a socket server that processes a gait cycle and outputs its deep feature vector. The vector is extracted using a deep neural network model, which is specifically trained based on provided training data. The default implementation is based on a Long Short-Term Memory network, which is trained on normalized and augmented gait cycles represented by time series of pre-selected angle and moment-power attributes of lower-limb joints.

The model can be replaced by specifying the path to a new trained model in: `src/main/classifiers/feature_vector/algorithm.py`. The model can even be retrained in case new training data are available, using the integrated training script: `train_model.py`.

<sup>2</sup>M. H. Schwartz and A. Rozumalski: The Gait Deviation Index: a new comprehensive index of gait pathology. *Gait&Posture*, 28(3):351-7, 2008. DOI: 10.1016/j.gaitpost.2008.05.001.

## 6 Comparison of Gait Cycles based on Time Series of User-Defined Attributes

This functionality allows users to compare gait cycles or aggregations of groups of gait cycles between each other.

### 6.1 How to compare visits?

First, pick the visits or gait cycles that you want to compare. Then, in the navigation bar, press the scales icon. You will be presented with groups of gait cycles that will be compared. Each group has its solid background filled with a random color. You can drag and drop gait cycles between groups as well as add new groups. After organizing the groups, you can select the set of attributes using which the groups are compared to each other in the form of a graph with time series.



Figure 7: The compare page – organization of groups

### 6.2 Add gait cycle(s) to compare

Find a visit in the patient's details page (Figure 4) you want to compare. In the visit detail section (Figure 5), press *Compare* on each gait cycle that you want to analyze. Each such gait cycle is added into a new group. By pressing

Compare all, all gait cycles from the given visit are added to the same new group.

### 6.3 What does the graph-based output represent?

After selecting the attributes, the graph will display the **average** value of the gait cycles in a particular group, with the point color same as of that group (Figure 8). The  $x$  axis shows the time, the  $y$  axis shows the values. In other words, a single time series is generated for each selected attribute and each group – the length of time series equals to the gait cycle length (i.e., the fixed length of 101 frames) and the values of the time series correspond to the average values calculated over all gait cycles belonging to the given group with respect to the selected attribute.

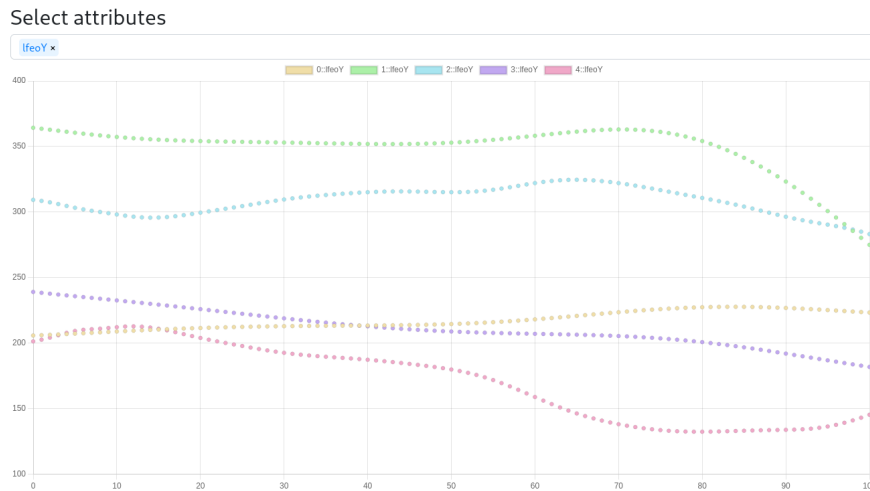


Figure 8: Comparison graph

## 7 Searching for Similar Gait Cycles

The main system feature is to find visits with *similar* gait cycles as a query visit. The query is compared against each database visit by computing the distance that expresses their dissimilarity. The fundamental part is to determine the distance between the query and a specific database visit. Since both query and database visits can be generally represented by sets of gait cycles, the system compares every gait cycle from one visit with every gait cycle from the other visit and aggregates the distances to obtain a single resulting distance. The system can also filter the compared visits by attribute-like data such as age, tags, or visit date.

The system provides various algorithms to determine the distance between two gait cycles. There are also two aggregation algorithms to calculate the resulting distance. In particular, the system supports three similarity-search modes: ATTRIBUTES, DEEP FEATURES, and GAIT\_QUALITY. The first mode tells the system to compare gait cycles based on the similarity of time series of additionally user-specified attributes. The second mode tells the system to compare gait cycles based on the similarity of deep-feature vectors that are extracted using the feature vector extractor service for each gait cycle. The third mode determines the similarity of gait cycles based on the absolute difference between their assessed qualities extracted using the quality extractor service.

To compare two visits (i.e., query and database visits) represented by sets of gait cycles, the system calculates the distance between each query and database gait cycle and aggregates the distances using one of the following aggregation modes: MIN or AVERAGE. The first mode selects the best (minimum) distance from the distances calculated between all pairs of the compared gait cycles. The second mode calculates the average distance between all pairs of the compared gait cycles.

Figure 9: Similarity form

## 7.1 How to find similar visits?

Open the visit details and press the *Find similar* button (Figure 5). There is the similarity form (Figure 9) where you need to specify the (1) similarity mode, (2) attributes along with a comparison algorithm in case of the selected

Similarity mode

ATTRIBUTES

Attributes

Choose Attributes...

Algorithm

-- pick algorithm --

Figure 10: Additional fields for ATTRIBUTES mode

ATTRIBUTES mode in the previous step (Figure 10), (3) aggregation mode, and (4) optional attribute-based filters (e.g., visit date range, patient age range, assigned visit's or patient's tags Figure 7). By clicking the *Submit* button, the similarity between the query and each database visit is determined. The visits are sorted from the most to the least similar with respect to the query Figure 11, which is quantified by a dissimilarity distance. Each result shows the visit's date, patient's identifier, number of associated gait cycles, the dissimilarity distance, and a classification score. The classification score represents the difference between the quality of the given visit and the subsequent visit of the same patient: green color indicates that the quality of the subsequent visit is better, red color indicates that the quality is worse, and N/A represents that no subsequent visit is available.

Result

Found 196 results

2022-10-09	Winna Arlet	5	0.00	N/A	first time important emergency
2010-12-10	Jmeno5000190 5000190	5	4.15	N/A	
2012-03-06	Jmeno5000194 5000194	5	7.18	-0.07	recording first time
2022-10-22	Anabella Frediani	3	40.33	-0.19	recording important emergency
2012-09-20	Jmeno5000102 5000102	5	79.86	N/A	recording first time emergency
2012-01-31	Jmeno5000113 5000113	5	81.92	N/A	emergency
2013-01-23	Jmeno5000018 5000018	5	82.07	-0.28	
2022-10-24	Larry Drivers	5	83.56	N/A	important
2022-10-06	Larry Drivers	5	84.04	-0.01	first time important emergency
2012-11-16	Jmeno5000034 5000034	4	84.45	N/A	dangerous

« 1 2 3 »

Figure 11: Result of the similarity search

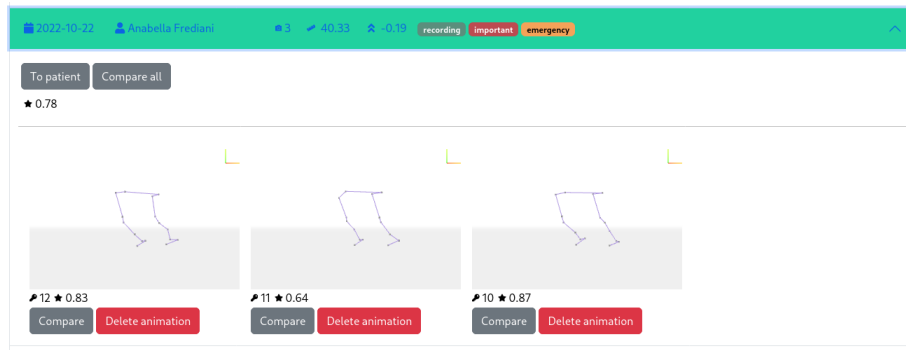


Figure 12: Visit's detail section in similarity result

## 8 Tags

Tags are colored labels that can be defined and associated with visits or patients (Figure 13), and used for attribute-based filtering during the process of searching for similar visits. In the navigation bar click on *Settings* (Figure 14). The system supports two types of tags – on the level of patients and visits. You can create, update, and delete the tags and also assign a custom color to each tag. When you edit a patient, you can specify their tags in the Tags field (Figure 15). A drop-down list will display possible tags. You can also start writing the partial tag's name and the list will filter matching tags. The tags on the level of visits can also be managed in the same way as on the level of patients (Figure 15).

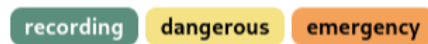


Figure 13: Tag list

Title	Last update	Action
Calculate deep features	10/22/2022 10:10:52 PM	<a href="#">Calculate</a>
Calculate quality	10/22/2022 10:11:11 PM	<a href="#">Calculate</a>

Patient tags		Visit tags	
<a href="#">first</a>	<a href="#">New tag</a> Edit Delete	<a href="#">important</a>	<a href="#">New tag</a> Edit Delete
<a href="#">second</a>	Edit Delete	<a href="#">recording</a>	Edit Delete
<a href="#">third</a>	Edit Delete	<a href="#">emergency</a>	Edit Delete
<a href="#">before last</a>	Edit Delete	<a href="#">first time</a>	Edit Delete
<a href="#">last</a>	Edit Delete	<a href="#">dangerous</a>	Edit Delete

Figure 14: Tag management

Tags

[second](#) × [first](#) ×

- [first](#)
- [second](#)
- [third](#)
- [before last](#)
- [last](#)

Figure 15: Tag auto-complete drop-down list

## 9 Appendix

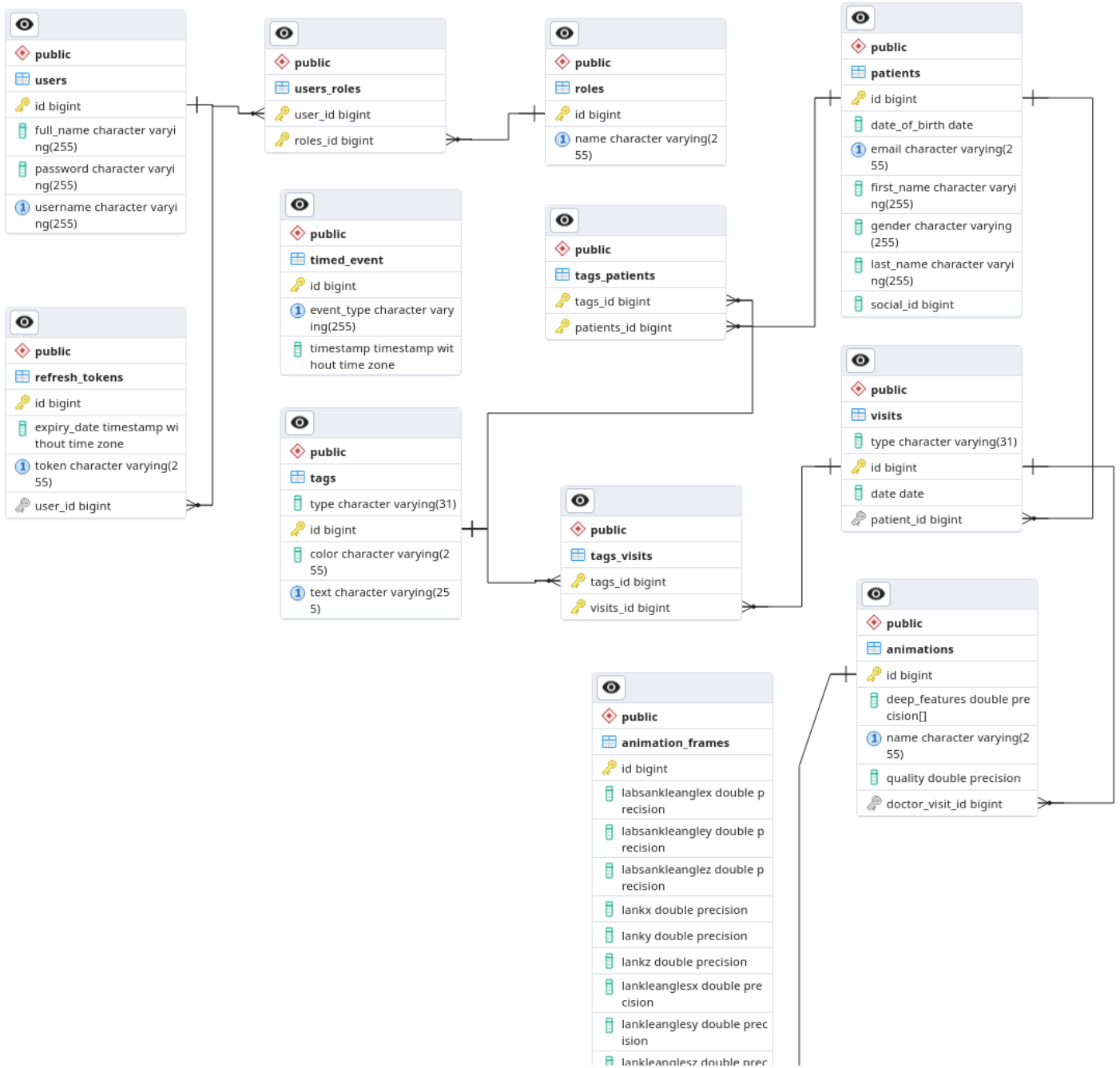


Figure 16: Database schema