

Abstract

Database architecture

SQL

- Accounts
- csv_file
- csv_log
- downsample
- downsample_group
- export
- feature
- feature_mapping
- imported_file
- imported_log
- imported_log
- influx_cluster
- medical_downsample
- medical_downsample_group
- medication
- patient
- version

influx

Import function

- Import EEG data manually
- Generating and importing EEG data automatically
- Import metadata
- Import progress check

Export function

Basic concepts

- Down-sampling and aggregation
- Aggregation groups
- Order of aggregation and down-sampling
- Start time, duration and down-sampling interval
- Insufficient and minimum valid data
- Time before/after medicine

Generate general query

Generate medication query

Generate aggregation groups

Export job

Progress checking

Visual data exploration

- Overview of medication and EEG graphs

Data management

Data validation

- Overview chart

- Detailed table

Timestamps in this system

- Timestamp of EEG record

- Test time of each CSV file

- Arrest time of each patient

- Chart time of every medication record

Error detection

Operation commit or cancel

- Database structure

- Code running process

Version control

- Design

- Export function

- Import function

- User management**

- Register and log in

- Profile page

- User management page

- Crash recovery**

- Import

- Auto-import

- Export

- Data maintenance**

- Database backup

- Log cleaning

- Export job cleaning

- System maintenance**

- Prerequisite for the development

- JAVA

- Integrated Development Environment(IDE)

- MySQL database

- Influx database

- Project codes

- Deploy the project on the server-side

- Restart the influxDB on the server-side

- Exceptions handling

- Add new administrator to the system

Abstract

BrainFlux is a data warehouse developed to manage large amounts of highly multivariate time-series data derived from biomedical applications. In its present form, it is optimized for handling of featurized electroencephalographic (EEG) data. BrainFlux includes three major functions: data import, data management, and data export. This manual summarizes the implementation and logic of each function, and provides additional technical details relevant to system maintenance and use.

Database architecture

Before dive into the functions of the system, you need to understand the architecture of the databases, in other words, understand how we organize data in our databases.

SQL

we use relational database mainly for three reasons:

(1) Store metadata of patient such as "patients" table which stores basic patients' information and "medication" table which stores patients' medication records

(2) Store imported files' information that are used for navigate influx database, for example, "csv_files" table contains basic summary of each imported files that are in the influxDB, we then use them to generate queries for EEG data.

(3) Store information that are used for system maintenance such as "version_control" table are used for maintaining database in different versions, "Accounts" table are used for access control.

The detailed stucture of SQL database is described below:

Accounts

This table stores information for registered users and administrators, most of the information can be changed from users' profile page.

field name	explanation
id	the auto incremental primary key
username	username that is used for login
email	user email
password	encoded password
role	distinghish users from administrators
first_name	user first name
last_name	user last name
enable	administrators have authorities to enable/disable users
last_update	latest time when this record is modified
create_time	the time when this record is created
database_version	the database verion that this user is using when export

csv_file

This table stores information for imported files, make sure information in this table is consistent with it in influx database so that we can correctly generate influx query from this table.

field name	explanation
id	the auto incremental primary key
pid	patient id
filename	name of this file
start_time	the time of the first EEG record in this file
end_time	the time of the last EEG record in this file
length	number of records in this file
density	the number of actual records divide by the maximum number of possible records during the same time range
machine	from which machine this file is uploaded
ar	binary number denote whether this file is processed using Artifact Reduction
path	the upload directory of this file
size	size of this file
uuid	unique id for a pair of files (ar / noar)
header_time	The time when this file is created, which is recorded in the header of the file (test date & test time)
last_update	latest time when this record is modified
conflict_resolved	binary number set by administrators denote whether or not all the errors detected from friend-end should be ignored
status	see "Data management / Operation commit or cancel" section for details
comment	users' comment for this file
start_version	see "Version control / design" section for details
end_version	see "Version control / design" section for details

csv_log

This table archive all the operations conducted to the csv files, right now only used for recovery from failed import/deleted, see "Data management / Operation commit or cancel" section for details.

field name	explanation
id	the auto incremental primary key
status	the status of this CSV file when it is recorded in this log
activity	the action of the manager for this file (pending commit, start commit, finished commit, cancel, start cancel, finished cancel)

field name	explanation
timestamp	the time of this action
filename	name of this file
start time	record start time
Endtime	record end time

downsample

This table stores export setting that will then be used for creating export job, see "Export function / basic concepts " section for details.

field name	explanation
id	the auto incremental primary key
alias	user-defined name for this export setting
peroid	downsample peroid setting

| duration | the export time duration for each patient
 | origin | | min_bin | minimum number of records required for export, the system will ignore this patient if don't have enough data | min_bin_row | | downsample_first | binary number indicating aggregation first or downsample first when export using this setting | create_time | the time when this setting is created | update_time | the time when this setting is last updated | deleted | binary number indicating whether this setting is deleted or not |

downsample_group

This table stores user-defined features groups they want, each one of them is associate with one query in "downsample" table.

field name	explanation
id	the auto incremental primary key
query_id	foreign key to "downsample" table, which associate export job to the desired query setting
label	user-defined name of the downsample group
downsample	downsample method
aggregation	aggregation method
columns	corresponding mapping to the columns in the raw data
create_time	the time when this setting is created
update_time	the time when this setting is last updated
deleted	binary number indicating whether this setting is deleted or not

export

This table stores exported jobs information,, see "Export function" section for details.

field name	explanation
id	the auto incremental primary key
query_id	foreign key to "downsample" table, which associate query setting to the desired output features
ar	binary number denote whether this file is processed using Artifact Reduction
layout	
patient_list	list of patients that expected to export in this job
finished	binary number denote whether this job is finished or not
canceled	binary number denote whether this job is canceled or not
failed	binary number denote whether this job is failed or not
machine	only export files imported from specified machine
db_type	
db_version	the database version which this job exported from
query_json	json format information of query setting
create_time	the time when this setting is created
deleted	binary number indicating whether this job is deleted or not
medication	binary number indicating whether this job is medication query or not
username	the machine where this job is created
finished_patient	number of patient finished export
all_patient	number of patient that is exported from this job

feature

This table stores information about measurements and electrodes.

field name	explanation
id	the auto incremental primary key
type	measurement name
electrode	electrode name
freq_low	
freq_high	
brain_location	location of the electrodes
notes	extra information for this record
SID	corresponding column name in the raw data
SID_Count	number of electrodes used for this measurement
csv_name	
csv_id	

feature_mapping

This table stores mapping information between electrodes and columns in the raw data.

field name	explanation
id	the auto incremental primary key
suffix	suffix of the electrodes
range_low	
range_high	
type	measurement name
comment	extra information for this record

imported_file

This table stores information about imported files, which is then used for avoiding duplicate import.

field name	explanation
id	the auto incremental primary key
filepath	file directory where this file is uploaded
filename	name of this file
filelines	number of lines in this file
filesize	size of this file
PID	the patient id associate with this file
isAr	binary number denote whether this file is processed using Artifact Reduction
timestamp	the time when this file started importing
uuid	unique id for a pair of files (ar / noar)
deleted	binary number denote if this file is deleted from influx database
delete_time	

imported_log

This table records log of import progress, and this table will automaticlly add one record every 15 seconds.

field name	explanation
id	the auto incremental primary key
uuid	unique id for a pair of files (ar / noar)
batch_id	id of this importing job
filename	name of this file
status	status of the import progress
timestamp	the time when this file started importing
all_percent	percent of the files that has been imported
this_percent	percent of the line in the current file that has been imported
all_file_size	total size of all the files is this batch
this_file_size	size of thi file
reson	store reason when this file imported failed
lost	binary number denote if there are lines skipped when imported

imported_log

This table records status of all the importing / imported files, which will be show in the front-end, and this table is updated using sql trigger of "import_log" table

field name	explanation
id	the auto incremental primary key
uuid	unique id for a pair of files (ar / noar)
batch_id	id of this importing job
filename	name of this file
status	status of the import progress
reson	store reason when this file imported failed
create_time	the time when this file started importing
update_time	the time when this record is updated
lost	binary number denote if there are lines skipped when imported

influx_cluster

This table stores information of machines

field name	explanation
id	the auto incremental primary key
cluster_id	
machine_id	id of the import machine
ip_addr	ip address of this machine
fail_path	move the raw .csv files to this path if imported failed

medical_downsample

Similar to the "downsample" table, this table stores medication export setting that will then be used for creating medication export job, see "Export function / Generate medication query " section for details.

field name	explanation
id	the auto incremental primary key
alias	user-defined name for this export setting
medication	name of the medication this query is asking for
before_medication	time before each medication that will be considered
after_medication	time after each medication that will be considered
peroid	downsample peroid setting

| duration | the export time duration for each patient
 | origin | | | min_bin | minimum number of records required for export, the system will ignore this patient if don't have enough data | | min_bin_row | | | downsample_first | binary number indicating aggregation first or downsample first when export using this setting | | create_time | the time when this setting is created | | update_time | the time when this setting is last updated | | deleted | binary number indicating whether this setting is deleted or not | | data_before_medicine | binary number denote whether should consider EEG data before first medication | | only_first | binary number indicating whether should only consider EEG data around first medication |

medical_downsample_group

Similar to the "downsample" table, this table stores user-defined features groups they want, each one of them is associate with one query in "medical_downsample" table.

field name	explanation
id	the auto incremental primary key
query_id	foreign key to "downsample" table, which associate export job to the desired query setting
label	user-defined name of the downsample group
downsample	downsample method
aggregation	aggregation method
columns	corresponding mapping to the columns in the raw data
create_time	the time when this setting is created
update_time	the time when this setting is last updated
deleted	binary number indicating whether this setting is deleted or not

medication

This table stores all the medication records information such as dose and unit.

patient

This table stores information of patients, each patient has more than 5000 features.

version

This table stores information of published database summary, see "version control" section for details

field name	explanation
version_id	the auto incremental primary key
create_time	time when this version is published
patient_num	number of patients that are available in this version
csv_file_num	number of .csv files that are available in this version
medication_num	number of medication records that are available in this version
csv_increase	number of .csv files that are added since last version
csv_delete	number of .csv files that are deleted since last version
comment	comment for this version
PUH_patients	number of "PUH" patients
UAB_patients	number of "UAB" patients
TBI_patients	number of "TBI" patients
patient_with_csv	number of patients with EEG data that are available in this version
total_length	number of hours of EEG data stored in the influx database
db_size	influx database size

influx

influx database is used to store time-series data, in our case, the EEG data of patients. Each patient has one unique measurement labeled by their patient id, and we use tags to differentiate segments of the data which .csv is import from:

tag name	explanation
arType	binary number denote whether this file is processed using Artifact Reduction
fileName	name of the .csv file which this record comes from
file_UUID	uuid of corresponding file

Import function

The first step of building a data warehouse is to import data into the database. Typically, source data are stored in a semi-structured form, but different data elements (e.g. EEG data and medication data, or EEG data from different subjects) may existed in different files or with different time-stamps. Importing serves to aggregate data and aligns data from different sources to a common timestamp. Data are imported into BrainFlux from .csv files..

Import EEG data manually

Data can only be imported into BrainFlux locally (i.e. when stored on the iMac Pro that hosts the database), so although the import function can be accessed from any computer with internet connectivity, files cannot be imported when the function is accessed through a remote device.

Presently, most data imported into BrainFlux are stored in .csv files exported by Persyst (<https://www.persyst.com/>). Raw EEG data typically sampled at 256Hz undergo featurization in the Persyst software environment. After featurization, the user may manually select some or all features for export. Our present panel of features summarizes various characteristics of the raw EEG waveform including component frequencies, amplitudes and rhythmicity, and export generates a .csv file with approximately 6,000 columns (LINK TO EEG CODEBOOK HERE). EEG features exist at a sampling frequency of 1Hz. These features can be exported with (AR) or without (NOAR) automated artifact reduction

.CSV file names contain several important data elements that are extracted on import to index the new file correctly. An example file name is 'PUH-2015-123_01ar'. Included are 1) the deidentified patient ID (PUH-2015-123), which links the EEG data to time-invariant patient meta-data; 2) an arbitrary serial number (01), which distinguishes files derived from consecutive original EEG recordings; and 3) an indication whether the file was exported with or without artifact reduction (ar).

Within the .csv file itself, several pieces of information are contained within the header rows.

File	F:\failed -- re-export\PUH-2015-015_02\Xxxxxxx- Xxxxx_76d48314-5658-4e53-8d21-2409f090d2fc.erd								
PatientName	Xxxxxxx	Xxxxx							
PatientID									
PatientBirth	1/1/1944								
TestDate	2015.01.28								
TestTime	3:01:58								

These include the file path of the source EEG recording (row 1); placeholders for the subject's name, medical record number and date of birth (rows 2-4) that are x-ed out during the deidentification process; and the timestamp at which the source EEG recording was created (rows 5-6). NOTE, this is the timestamp at which the EEG technician first created the new recording, but is antecedent to the actual start time of data because there are some additional tasks completed by the EEG technician after creating the study but before data are recorded. Also, the file path contained in row 1 contains a unique string of characters corresponding to the deidentified source EEG file. These are reproducibly generated by Natus NeuroWorks at the time of deidentified export, such that re-exporting the same raw EEG file will generate the same string of characters. This character string does not vary based on whether data are exported with or without artifact reduction, so uniquely identifies the source file. We term this string "UUID" (unique id).

In order to import new EEG data from .csv files generated by Persyst, the user needs to log into BrainFlux as an administrator, navigate to the "Raw Data / Import CSV" page, enter the correct directory path to identify where the .csv files are stored, and hit "search folder" button. All .csv files contained in the the target directory will populate on the import page, and the user may select some or all of the files to import, then execute the import task by clicking "Import CSV files."

The code of import function is in the "ImportCSVService.java". In order to increase the import speed, the import function is a multi-thread process. The import function will first read the header to extract useful information such as patient id, header time, and uuid, which are later be stored in "csv_file" table as file metadata. The system will then perform several data integrity checks (detailed below). If all checks are passed, the system will then read the records line by line and write into influxDB in batches. Upon finishing the import process, the system will add one record to 'csv_log' and 'imported_files' tables respectively, and then move the source file to a stable directory depending on whether it's imported successfully or not.

One special circumstance is created when a source EEG file is too large for Persyst to export as a single .csv (this typically occurs when source recordings last more than approximately 35 hours). In this situation, the user will have had to export approximately half of the features in one .csv (noted "PUH-2015-123_01ar-A") and the second half of the features in a second file (noted "PUH-2015-123_01ar-B"). We refer to these files as "OOM" files (out of memory). The OOM files have fewer columns than ordinary files individually, but it's the same number of features when all the component OOM files associated with the same file name are concatenated. OOM files are combined into one record in the Influx database.

There are several types of errors will stop the import progress for a file, when one of the following 6 errors happened, the system would not import this file and write a log in the "import_log" table, along with the reasons:

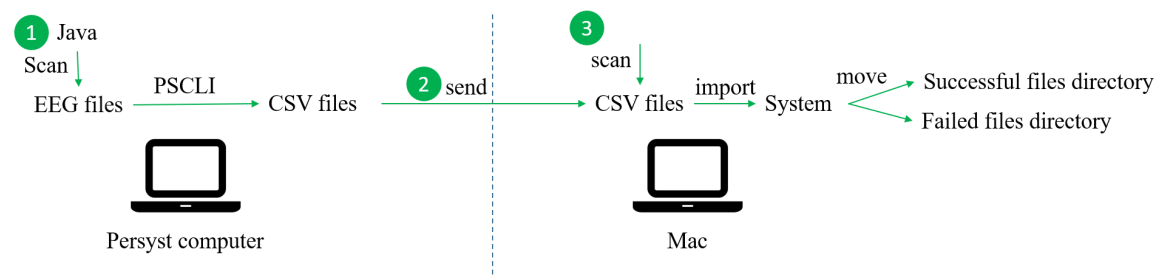
- (1) Patient ID in the first line of the csv is different from the ID in the file name
- (2) File does not have a valid UUID.
- (3) The file type (ar / noar) is not specified in the file name.
- (4) A file generated from the same source EEG (i.e. same UUID) with the same AR/NOAR designation was previously imported and never deleted. This is determined by searching the "imported files" table to determine if there is a matching UUID with the same AR/NOAR designation and "Deleted" feature of 0 (see below for an explanation of the "deleted" feature).
- (5) The system detects the corresponding file lock exists in the same directory which means this file is being imported by another thread, the current thread should stop and find another file. This may also happen when the system crashed while importing this file before.
- (6) The different parts of the OOM files were imported in the previous version. Users should delete all the OOM files associated with this file name before re-import.

When following errors exist in the certain line of the importing file, the system will skip this line, and write a log in the "import_log" table, along with the reasons:

- (1) The number of the columns in this line is inconsistent with the number of features in this file.
- (2) Measurement time of this line is earlier than the test start time written in the file header.
- (3) Measurement time is the same as the previous record.

Generating and importing EEG data automatically

The following figure demonstrates the basic process of auto-import, which is consist of 3 steps:



(1) At certain time of the day, the JAVA program on the windows computer on which Persyst is installed will scan the pre-defined directory where patient EEG files waiting for featurization and csv export are stored. The folder structure of this directory is: "~\Waiting_to_processes\Patient ID /recording number/files/" (for example "~\Waiting_to_process\PUH-2019-001/ PUH-2019-001_01/" , where the ~\PUH-2019-001_01/ folder contains the original de-identified EEG recording as exported from Natus NeuroWorks (the electronic medical record). When EEG files are identified in

this parent directory, the JAVA program will launch the Persyst-based featurization process using a batch process function to control Persyst. After all featurization (signal processing) is complete, the JAVA program will generate the appropriate CSV files with- and without- artifact reduction using the PSCLI (Persyst Command-Line) interface. These .CSV files will be stored in the "/ready_to_send/" directory if successfully generated.

(2) The JAVA program then uses SCP to send these CSV files to a receiving target folder on the Mac computer that hosts BrainFlux, and also sends an empty .txt file for each .CSV file with the same name as an indicator when files are completely transmitted.

(3) At another time of the day, the website deployed on the Mac computer will scan the receiving folder, if it finds the .csv files and their corresponding .txt files with the same name, it will call the import function described in the previous section.

The corresponding function in the Persyst computer is "AutoImportEEG" project, which will run automatically when the computer is rebooted, and the automatically scan time currently is 12am. We use following command line to process .erd files (.erd is the file format of Natus-generated EEG waveform data), Users can find detailed description of all the options available using `PSCLI.exe /?` under " /Persyst / insight" directory:

```
'C:\Program Files (x86)\Persyst\Insight\PSCLI.exe /Panel=\"%s\"  
/SourceFile=\"%s\" /OutputFile=\"%s\" /MMX=\"%s\" /Process /ExportCSV''
```

The import function on the BrainFlux iMac Pro host computer is in the 'AutoImportServices.java', which will scan the files from Persyst computer if .csv files and corresponding .txt files are found, the system will call the import function from "ImportCSVService.java", and move all the imported files to other folders when finished.

Import metadata

Time-invariant patient metadata are stored by the clinical Post-Cardiac Arrest Service (PCAS) using a registry built in REDCap. This includes approximately 1000 data elements per patient (<https://bf.tsdb.science/manual/EEGDataCodebook>). Only de-identified data are transferred into BrainFlux, and these are entirely redundant with the data stored in REDCap. The purpose of this duplication is to allow data exploration within BrainFlux using filtering based on metadata (e.g. review EEG data only of survivors to hospital discharge). These metadata are saved in the MySQL table called "patient".

Similar to importing the EEG .csv files, the user needs to type in the directory of the metadata .csv file and import. The system will read the .csv file line by line and write them into the SQL database.

Import progress check

A user can check the progress of a large-scale import of .csv EEG files using the "Raw data / Import activity" page, where all the ongoing jobs are shown on the top. The user can click on specific batch import id to see the detailed information, there is also a progress bar on the bottom of the page, where all the files in queue will be shown, along with overall import progress. Notice that the ongoing files' bar is blue, questionable files' bar is red and finished files' bar are green. This progress page will refresh every 15 seconds.

The progress is consistent with "ImportProgress" table. Every 15 second, The java code in "ImportProgressServices.java" will add a record into the 'import_Log' table, the corresponding record in "import_progress" table will be automatically updated by SQL trigger associated with "Import_Log" table. Every 15s, the system would calculate the percentage of finished part and add

a new log. The system would also update this percentage into progresss tabel.

Export function

Featurized EEG data exists in Influx at a frequency of 1Hz. These data can be exported with or without additional summarization for the purpose of exploration or analysis using other software platforms. The user has considerable flexibility to define custom exports that summarize subsets of the data that are of particular interest for any given application.

Basic concepts

Before building the query, users should understand some basic concepts which will be used later.

Down-sampling and aggregation

When less granular data are desired for analysis, this may occur by summarizing data over time (we refer to this as "downsampling," for example summarizing the median value of a particular feature every hour) or across features (we refer to this as "aggregation," e.g. summing the bandpass filtered spectral power across several adjacent electrodes).

The following image shows an example of a BrainFlux source spreadsheet illustrating two data axes. The horizontal axis contains different qEEG features while the vertical axis represents data summarized over consecutive seconds.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
7	Artifact Intensity																	
8	ClockDateTime	Time	I1_1	I1_2	I1_3	I2_1	I2_2	I2_3	I2_4	I2_5	I2_6	I2_7	I2_8	I2_9	I2_10	I2_11	I2_12	I2_13
9	40396.80436	0	0	0	0	0.00572	0.00572	0.00572	0.00572	0.00572	0.00572	0.00572	0.00572	0.00572	0.00572	0.00572	1	0.00572
10	40396.80438	1	0	0	0	0.034623	0.028893	0.090001	0.008542	0.005639	0.010179	0.018445	0.012196	0.005639	0.016364	0.014326	1	0.005639
11	40396.80439	2	0	0	0	0.003867	0.003867	0.003867	0.003867	0.003867	0.003867	0.003867	0.003867	0.003867	0.003867	0.003867	1	0.003867
12	40396.8044	3	0	0	0	0.00195	0.00124	0.003562	0.001201	0.002318	0.003071	0.001201	0.001485	0.001264	0.001201	0.001201	0.930712	0.00204
13	40396.80441	4	0	0	0	0.112522	0.023546	0.013589	0.020128	0.074274	0.071356	0.009461	0.012678	0.015849	0.009048	0.011839	1.00086	0.022966
14	40396.80442	5	0	0	0	0.003932	0.005814	0.007027	0.001063	0.001726	0.003279	0.005158	0.00298	0.001063	0.004175	0.006461	0.989942	0.002888
15	40396.80443	6	0	0	0	0.003325	0.007126	0.006626	0.001062	0.001761	0.002714	0.005685	0.001062	0.001062	0.002776	0.00977	0.983531	0.002117
16	40396.80444	7	0	0	0	0.004214	0.002967	0.003651	0.001693	0.002589	0.003598	0.003064	0.00189	0.000613	0.001883	0.00171	0.96449	0.003866
17	40396.80446	8	0	0	0	0.008306	0.004488	0.005495	0.003501	0.00472	0.005736	0.004756	0.001536	0.001304	0.004976	0.008003	0.995111	0.006431
18	40396.80447	9	0	0	0	0.001877	0.005533	0.006269	0.0073	0.010144	0.00664	0.004749	0.003842	0.001314	0.005595	0.011016	0.999027	0.009879
19	40396.80448	10	0.375129			0.00134	0.003827	0.005497	0.00652	0.007297	0.005615	0.004929	0.004498	0.000583	0.006253	0.005419	0.996916	0.010831
20	40396.80449	11	0.375129			0.001593	0.002838	0.004353	0.004874	0.01007	0.004691	0.004255	0.004212	0.001593	0.005515	0.004238	0.990902	0.004989
21	40396.8045	12	0.375129			0.013164	0.013164	0.013164	0.013164	0.013164	0.013164	0.013164	0.013164	0.013164	0.013164	0.013164	0.776724	0.013164
22	40396.80451	13	0.375129			0.004197	0.003999	0.00584	0.01096	0.004989	0.00899	0.005959	0.001773	0.000597	0.004286	0.005332	0.062943	0.003828
23	40396.80453	14	0.375129			0.006337	0.006337	0.006337	0.006337	0.006337	0.006337	0.006337	0.006337	0.006337	0.006337	0.006337	0.006337	0.006337
24	40396.80454	15	0.375129			0.00203	0.002902	0.001792	0.002304	0.002603	0.003236	0.003528	0.00318	0.000755	0.00338	0.003648	0.000755	0.003008
25	40396.80455	16	0.375129			0.004031	0.00279	0.004658	0.003899	0.003383	0.001701	0.004061	0.000799	0.003706	0.004548	0.004148	0.000799	0.004406
26	40396.80456	17	0.375129			0.001237	0.003154	0.004046	0.001237	0.004017	0.001237	0.002682	0.001237	0.001237	0.001462	0.003609	0.001237	0.002276
27	40396.80457	18	0.375129			0.002129	0.002637	0.002487	0.001381	0.002417	0.001207	0.002786	0.001207	0.001207	0.001207	0.003038	0.001207	0.002654
28	40396.80458	19	0.375129			0.001032	0.002116	0.003461	0.002671	0.004166	0.003507	0.004404	0.001032	0.002007	0.004066	0.004155	0.001032	0.003635
29	40396.80459	20	4.23507			0.000634	0.001814	0.004161	0.00207	0.0029	0.004522	0.004504	0.002204	0.001135	0.003371	0.004256	0.000634	0.003045
30	40396.80461	21	4.23507			0.003847	0.001805	0.00139	0.001599	0.002476	0.003504	0.003502	0.001985	0.000864	0.001827	0.003759	0.000864	0.003862
31	40396.80462	22	4.23507			0.00166	0.002102	0.003756	0.002683	0.002513	0.003	0.003728	0.00166	0.00166	0.001667	0.004503	0.00166	0.004448
32	40396.80463	23	4.23507			0.000607	0.002709	0.003976	0.002043	0.001694	0.003172	0.004151	0.001935	0.000607	0.001955	0.004477	0.000607	0.002554
33	40396.80464	24	4.23507			0.003008	0.001593	0.001628	0.002388	0.002351	0.002967	0.003548	0.000976	0.000976	0.000976	0.003086	0.000976	0.001327
34	40396.80465	25	4.23507			0.000688	0.002868	0.002645	0.004261	0.002847	0.003213	0.004623	0.003457	0.000688	0.003662	0.004857	0.000688	0.002026
35	40396.80466	26	4.23507			0.001903	0.001079	0.002008	0.004546	0.002291	0.003243	0.003749	0.00258	0.0021	0.003569	0.001853	0.000748	0.003712
36	40396.80468	27	4.23507			0.001419	0.001419	0.003034	0.0046	0.002873	0.00251	0.004811	0.001419	0.001419	0.003191	0.001419	0.001419	0.002551
37	40396.80469	28	4.23507			0.000963	0.001444	0.002852	0.003083	0.001985	0.00229	0.00414	0.001859	0.000963	0.002782	0.001983	0.000963	0.001059
38	40396.8047	29	4.23507			0.001429	0.000662	0.001876	0.002172	0.001917	0.002414	0.003941	0.001748	0.000662	0.003702	0.000662	0.002513	0.003404
39	40396.80471	30	4.19253			0.003599	0.002066	0.003621	0.002771	0.002534	0.003417	0.003471	0.002348	0.002066	0.002779	0.00372	0.002066	0.004766

Specifically, ggregation means to summarize the data by different attributes along the horizontal axis using operations such as SUM or MEAN.

Down-sampling means to summarize data over time (along the vertical line).

Aggregation groups

An aggregation group is a user-defined group of qEEG features, each of which is originally contained in single columns (i.e. user-defined attributes) with that are to be summarized.

Order of aggregation and down-sampling

Since aggregation is performed along the horizontal axis and down-sampling is performed along the vertical axis, the order of these two operations matters. Consider a simple data set with 2 columns (A1 and A2) and 3 rows. Assume that we use the MEDIAN down-sampling method, while the aggregation method is SUM. We will include A1 and A2 into the same aggregation group and down-sample all available data.

A1	A2
2	5
1	7
3	9

If we do the aggregation first then down-sampling, the result would be as follows:

$$Result' (/ * + = median(sum(2, 5), sum(1, 7), sum(3, 9)) = 8$$

On the contrary, if we down-sample first, the result would be:

$$Result * +/' (= sum(median(2, 1, 3), median(5, 7, 9)) = 9$$

The two results are different, therefore specifying the order of operation is important and is specified by the user.

Start time, duration and down-sampling interval

Start time is an offset from the initiation of EEG monitoring before which each patient's data should be ignored. For example, if the investigator were interested in analyzing data measured between 24 and 36 hours after initiation of monitoring, he or she would define the start time to be 24 hours. This applies to all subjects' data including in the export task.

The duration is the amount of time analyzed after the start time for each patient (how much data should be covered), and also applies to all patients in an analysis task and will override the original end time of data. In the example above, the user would select 12 hours to limit the data range from 24 to 36 hours. If the user-defined duration is longer than a particular subject has available data, "N/A" will be returned for time periods after no additional data are available (see below).

Down-sampling interval (or bin) is the time resolution to which data should be down-sampled.

Each row of the final result is grouped by this interval and down-sampled with the user-specified method.

Insufficient and minimum valid data

Minimum valid data is a threshold, patients with less available data than this minimum threshold will not be included in the data export result, and will be listed in the export output summary file instead. For example, if only subjects with at least 6 hours of available EEG data are to be included, subjects with less data available will be omitted from the export.

Time before/after medicine

The time before and time after the medicine is used in the medical query. They represent the time range in the medical query. For example, a subject received valproic acid at 10:00 am every day. If the time before and after the medicine is set as 30 minutes, the system would take this patient's EEG data from 9:30 am to 10:30 am then do the aggregation. This feature is developed to explore medication effects on the EEG signal, since medications are an important potential confounder in many analyses.

Generate general query

General query means only to use patients' EEG data and aggregate it as the user defined. The general query does not account for medication administration or other data sources. To build a general query or select an existing query, users can go to the "export/query builder" page. Users need to set the following attributes then click the create button to create a new query.

Create Query

Alias

Pattern Finding on aEEG

Downsample Interval	Starting Time	Duration
1Hours	0Hours	4Hours
Minimal records in a valid bin	Minimal valid bins for a patient	Choose first operation
15Minutes	50% of total bins	<div>AggregationDownsample</div>

Create

After a general query is created, users can add some aggregation groups for this query and do the export.

Generate medication query

Medication query is intended to aggregate subjects' EEG data centered on administration of a particular medication of interest. Users can go to the "export / medical query builder" page to create or edit medical queries. In order to generate a medical query, users need to set the following attributes and add some aggregation groups for export.

Basic Info

Alias

test 1

Medicine name

1/2 NS + KCL 20 MEQ

Downsample Interval

1 Minutes

Time before medicine

6 Minutes

Time after medicine

6 Minutes

Only show the result of first medicine record

no yes

Make sure eeg data before first medicine record

no yes

Minimal records in a valid bin

Minimal valid bins for a patient

Choose downsample first or aggregation first

Aggregation First Downsample First

Input minimal rows / interval percentage for a valid bin

Unit

0 count

Save

Export

Delete

Generate aggregation groups

There are several user-defined parameters which must be specified to create a new aggregation group. First, the user must input a free-text label. Then, the summary statistics (eg. mean, sum, median, variance) for the vertical downsampling and horizontal aggregation must be selected. After that, users should select the category of qEEG features of interest (e.g. FFT spectrogram), the electrode(s) to be aggregated, and the feature(s) of the qEEG category. These features vary across qEEG column type. For example, features of an FFT spectrogram vary across the frequency domain, while amplitude values (aEEG) are summarized as the maximum, minimum, median, 75th%ile and 25th%ile within each second. Collectively, these parameters define a single aggregation group. The user can create as many aggregation groups as necessary.

Create New Aggregation Group

Label

Fz-Av17 1

Downsample Method (Vertical)

Standard Deviation 2

Aggregation Method (Horizontal)

Mean 3

Choose Columns 4

aEEG

aEEG+ (0.16 - 25Hz) (LFF 1 sec, HFF 25 Hz, custom (off))

Artifact Intensity

Asymmetry EASI/REASI

Electrode Signal Quality

FFT Spectrogram

PeakEnvelope

Relative Asymmetry Spectrogram

Rhythmicity Spectrogram

Seizure Probability

Spike

Suppression Ratio

Choose Electrodes 5

Predefined Sets

* [X-Av] I45 ~ I63

* [Bipolar] I64 ~ I81

Single Electrodes

Left Hemisphere

Right Hemisphere

Cz-Av17

Pz-Av17

Fz-Av17

P4-Av17

C4-Av17

F4-Av17

P3-Av17

Candidate 6

Max

Min

Median

75% Percentile

25% Percentile

7 Final List

→

aEEG

Fz-Av17

Median

To eliminate possible human errors, the add button above now always clears the final list and renew the list with your current selection.



The detailed mapping between the column names and the measurements are described in the page "manual / EEG data codebook" (<https://bf.tsd.bscience/manual/EEGDataCodebook>).

Export job

After a query is generated and aggregate groups are added, users can click the export button on the page to export the result of this query. The system would generate the Influx command and execute this command in the influx database. Then, the result would be written into .csv files that are available for users to download.

Progress checking

Users can go to the "Export jobs" page to see all export jobs have been created. The progress attribute estimates how much of this job has been finished. Multiple threads are used to do an export. The number of patients is known when the job is created, each thread handles a single patient at a time and adds a count of 1 when that patient's data aggregation is complete and the output is written. We use the count divided by the number of patients to do the progress estimation.



ID	Alias	Start Time	Progress	Results
472	20+ trends for manuscript	7/17/2019, 6:13:08 PM	100%	Download
457	Data for ML experimentation	7/12/2019, 5:17:00 PM	100%	Download
456	Data for ML experimentation	7/10/2019, 8:34:02 PM	100%	Download

Showing 1 to 3 of 3 entries

[First](#) [←](#) [1](#) [→](#) [Last](#)

Once the job is finished, users can download it on this page. A completed export task generates several files that are downloaded together as a .zip archive. 1) The exported data in wide format; 2) the exported data in long format; 3) a export metadata file summarizing key portions of the export task such as start time, stop time, database status at the time of export, subjects not included in the export job and the reason for non-inclusion. We also include a version of the exported data where each patient is saved to his or her own file ("split version") that may be more usable when exporting large amounts of data (e.g. many megabytes per subject) because it avoids excessively large aggregated file sizes.

Visual data exploration

The ultimate goal of examining EEG data is to see the effect of treatment and discover patterns in the EEG data, therefore we implement this page for an individual patient, which shows the trend of patient's EEG data, and the relationship with medication that this patient has taken.

Overview of medication and EEG graphs

When users click on the 'show patient EEG & medication data' in the file validation chart, it will jump to the page where users can query medication usage and EEG information for this specific patient, and the corresponding patient id is shown on the top.

PATIENT MEDICATION INFORMATION

Detailed information for treatments.

Data Overview of PUH-2015-015

[← Return to Charts](#)

Add Drug

Show EEG

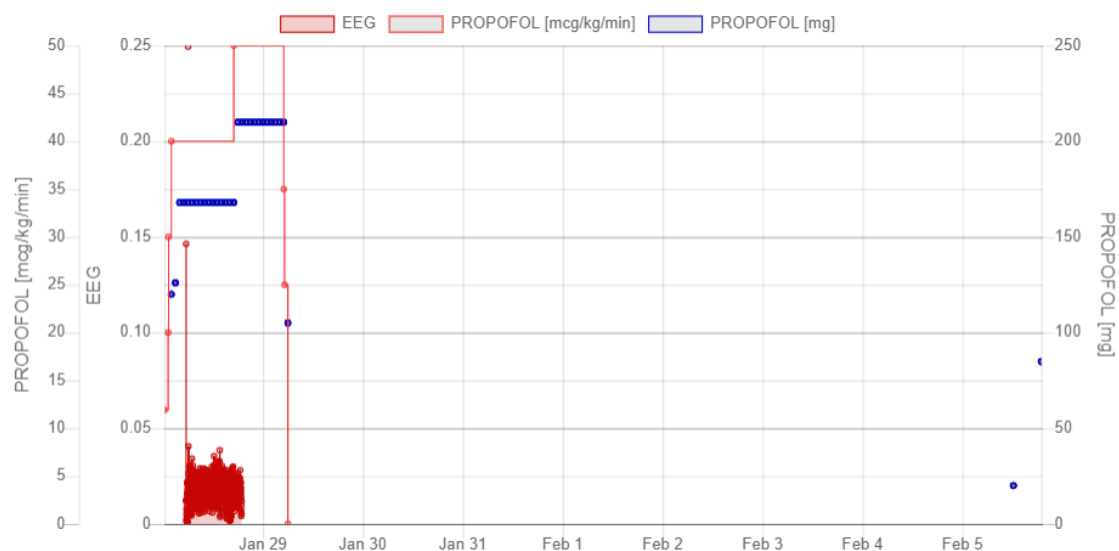
Go

Drug

remove

©2018 University of Pittsburgh. All rights reserved.

Users can choose medicines that this patient has taken, the records of dose and time will be shown in the graph after clicking the “Go” button, where the continuous line represents the continuous infusion, and the dots represent intermittent infusions.



When clicking on “add EEG” button, users can query the EEG data from inflexDB, as shown in the following figure, the interface of query EEG is similar to the interface in the Query Builder part, users can specify the Downsample and Aggregation method, AR or NOAR file they want to focus on, the columns and electrodes they select will then be interpreted to the corresponding columns in the origin patient file. The result of users’ queries for medication and EEG would be shown in an integrated chart, as well as the stacked version for each medication / EEG chart.

Edit Aggregation EEG

Downsample Method (Vertical)

Mean

Aggregation Method (Horizontal)

Mean

Choose downsample first or aggregation first

Aggregation
Downsample

Downsample Interval

1
Minutes

Minimal records in a valid bin

Input minimal rows / interval percentage for i %

Choose ar/noar file

AR
NOAR

Choose Columns

aEEG
aEEG+ (0.16 - 25Hz) (LFF 1 sec, HFF 25 Hz, custom (off))
Artifact Intensity
Asymmetry EASI/REASI
Electrode Signal Quality
FFT Spectrogram
PeakEnvelope
Relative Asymmetry Spectrogram
Rhythmicity Spectrogram
Seizure Probability
Spike
Suppression Ratio

Choose Electrodes

Predefined Sets

Single Electrodes
Fp1
F7
T3
T5
O1
F3
C3
P3
A1
Fz

Candidate

Electrode Signal Quality

Final List

Electrode Signal Quality
F7
T3
T5
O1
F3
Electrode Signal Quality

To eliminate possible human errors, the add button above now always clears the final list and renew the list with your current selection.

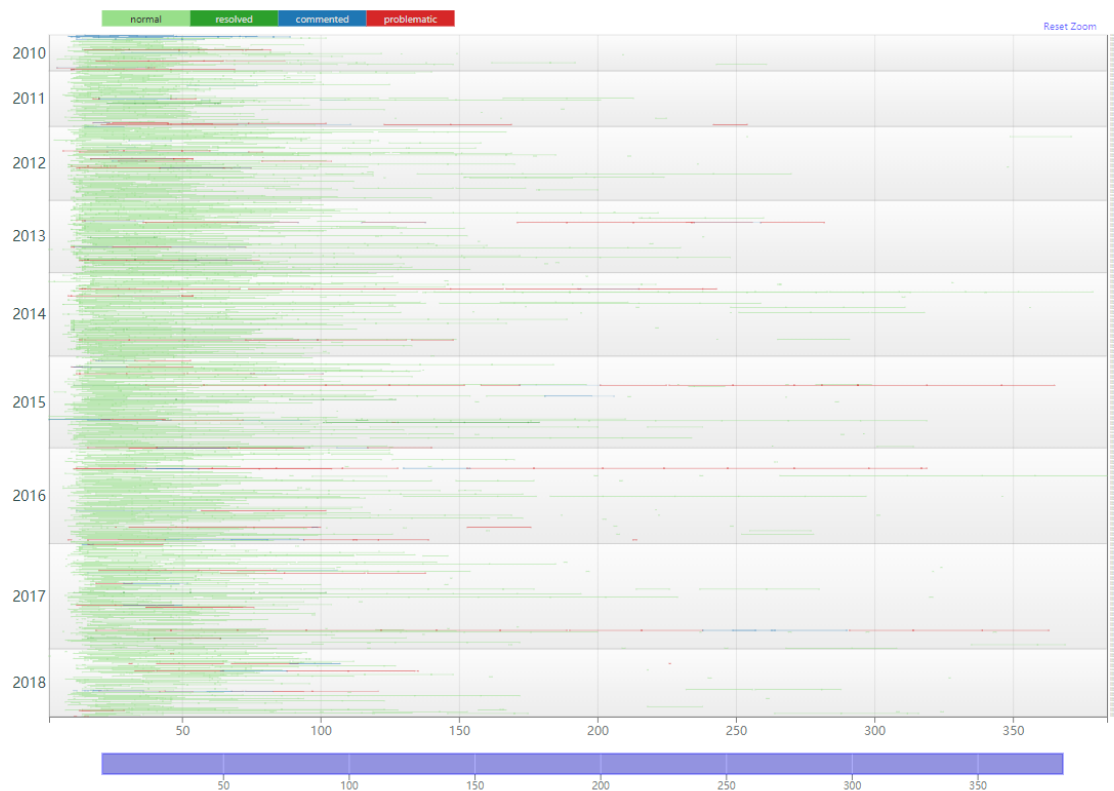
Data management

Another major function of the BrainFlux system is data management. The current system includes many terabytes of data derived from over a thousand patients, making it challenging for the user to have confidence in data integrity (missing data, corrupt data, etc). The data management function can help users to detect and fix errors quickly.

Data validation

In order to check the potential problems with the imported files, for example, overlap/ duplication of the EEG data, missing ar/ noar files, etc, we use a simple graph to visualize the overview of all the files in the database, and a table for detailed information of the individual file. Users can easily scan these graphical representations to detect and eliminate potential errors.

Overview chart



In the graph shown above, we use light green to denote normal files with no obvious error, use dark blue for files that have a comment from the users, use dark red to indicate problematic files, and the dark green represents files that users manually checked. The x-axis represents the time difference between the start time of the files and the arrest time of corresponding patients in hours, the origin time range is 360 hours, which is about 2 weeks, users can change the time range by adjusting the blue bar on the bottom of the graph. The y-axis on the left is the year and y-axis on the right contains their patient id and file type.

Users can zoom in the specific part of the graph by clicking and dragging. alternatively, clicking on the year label will show the patients' files in that year, and clicking on the legend on the top will show the files in the corresponding category. whenever users want to go back to the original scale, click the 'reset' button on the top right corner.

The missing files problem is easy to spot in the graph, the overlap of the EEG data will leads to darker red than its original color.

We also implemented a filter for users to select patients with special requirements, the graph will be refreshed according to the retrieved result. After using the filter, users need to click on "Overview of all patients" to go back to the original scale.

Add Filter

Go

Overview of all patients

Filter Field

=

Input value

remove

Detailed table

PUH-2010-061

Show patient EEG & medical data

Change the comment of this patient

All the files for the patient are resolved

Search for files in the table...

File Name	Start Time	End Time	Row Count	Density	Counterpart	File Gap	File Status	comment	Manage
PUH-2010-061_01_ar.csv	04/21/2010 13:34:46	04/22/2010 21:48:05	116000	1	Valid	11:14:46	Unsolved	Correct start time is 09:34:46 – time drift problem CHANGE	RESOLVED DELETE
PUH-2010-061_01noar.csv	04/21/2010 13:34:46	04/22/2010 21:48:05	116000	1	Valid	11:14:46	Unsolved	Correct start time is 09:34:46 – time drift problem CHANGE	RESOLVED DELETE

Showing 1 to 2 of 2 entries

After users clicking on the individual file bar or corresponding label on the right in the overview chart, the detailed information of that patient will be shown in the table. All the files considered as problematic will be marked as 'Unsolved' with a red background on the problematic field.

There are a few operations users can perform to the individual file: adding, changing or deleting comment on this file. when all the problems associated with this file are manually checked to be correct, marking the file as 'resolved' tells the system to ignore the detected problems with this file, this operation can also be done on patient level, then all the files associated with this patient will all be considered as 'resolved'. users can also delete files from this table, though it will be 'pseudo-deletion' according to the version control mechanism.

clicking on the green button below the patient id will lead to the EEG & medication page for this patient.

Timestamps in this system

There are several timestamps used in this system such as the timestamp of every EEG record, the test time of each CSV file, the arrest time of each patient, the chart time of every medication record. They are not in the same time zone. This part shows how the system handles the time zone problem.

Timestamp of EEG record

The timestamp of EEG record is a long integer which represents UTC in the CSV file. It is saved as UTC in both MySQL and Influx database. When it is shown on the frontend, it would be transformed into ET (Eastern Time). It would also be transformed into ET for export.

Test time of each CSV file

The test time of each CSV file which is also called as header time is eastern time. It is the time when this CSV file is created. Generally, the time difference between the header time and the timestamp of the first EEG record should not be longer than an hour.

Arrest time of each patient

The arrest time is when the patient is sent to ICU. It is also eastern time. This time should be earlier than any other timestamps of this patient.

Chart time of every medication record

In the MySQL database, medication records about every patient are saved. They include the name, dose, the time when the patient took this medicine. The timestamp of this record is also eastern time. For medical query which can export the aggregation result of EEG records around one medication record, the timestamp of EEG records would be transformed to ET.

Error detection

The page called "detect wrong data" is used to detect errors in the database. The following 5 different kinds of errors would be detected for each patient in this database.

Error	Description
Overlap	There are at least two .csv files for this patient have overlap in their time ranges
Missing ar	For this patient, these ar files are missing
Missing noar	For this patient, these noar files are missing
Different time range	For this patient, there are some ar or noar files have different time range with their counterpart
Wrong name	The names of some .csv files are not in the correct format

Patients with error would be marked as red in the overview chart. If the error detection table is not empty, a new version of this database cannot be published.

Operation commit or cancel

In the Brainflux system, new data should be imported and wrong data should be deleted and reimported. Operation commits or cancel is used to do the double-check to avoid the maloperation.

Database structure

Two tables in the MySQL database are used to achieve this logic. The first one is csv_file and the second one is csv_log. In the csv_file table, apart from the basic information about each CSV file, a column called status which is used to save the status of this file in the database. The following table shows the detail.

status No.	explanation
0	this file is in the stable database and should not be changed
1	this file has been deleted but not confirmed
2	this file has been imported but not confirmed

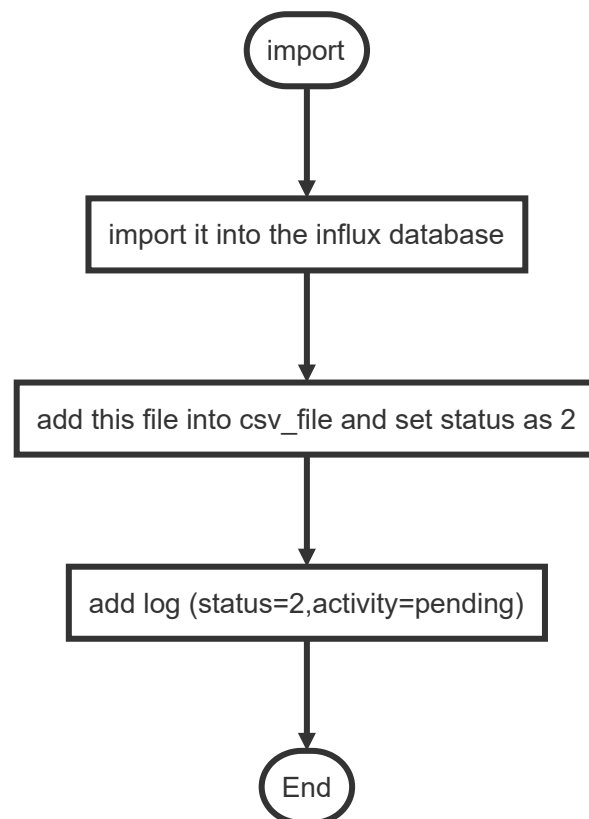
There is another column in csv_file called the timestamp which is used to record the latest altered time.

The csv_log table is used to save every action about the csv_file table. The following table shows the columns and explanation.

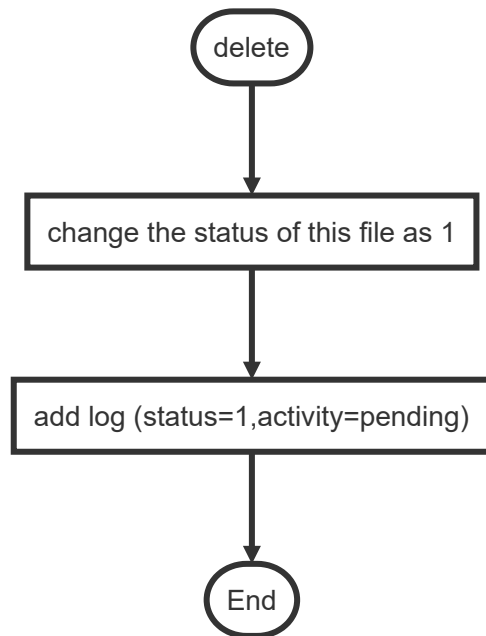
column name	explanation
id	the auto incremental primary key
status	the status of this CSV file when it is recorded in this log
activity	the action of the manager for this file(pending,commit_start,commit_finished,cancel_start,cancel_finished)
timestamp	the time of this action
filename	name of this file
start time	record start time
Endtime	record end time

Code running process

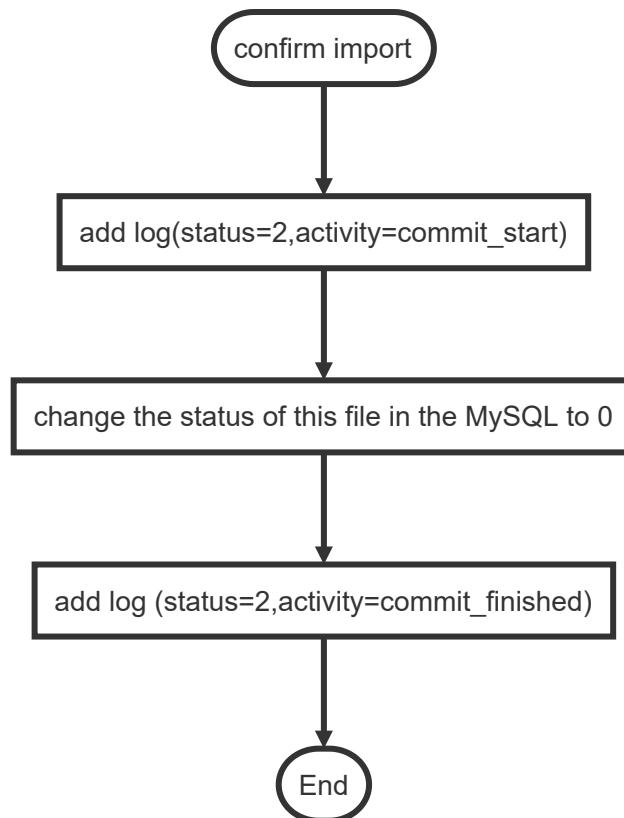
Import a new csv file:



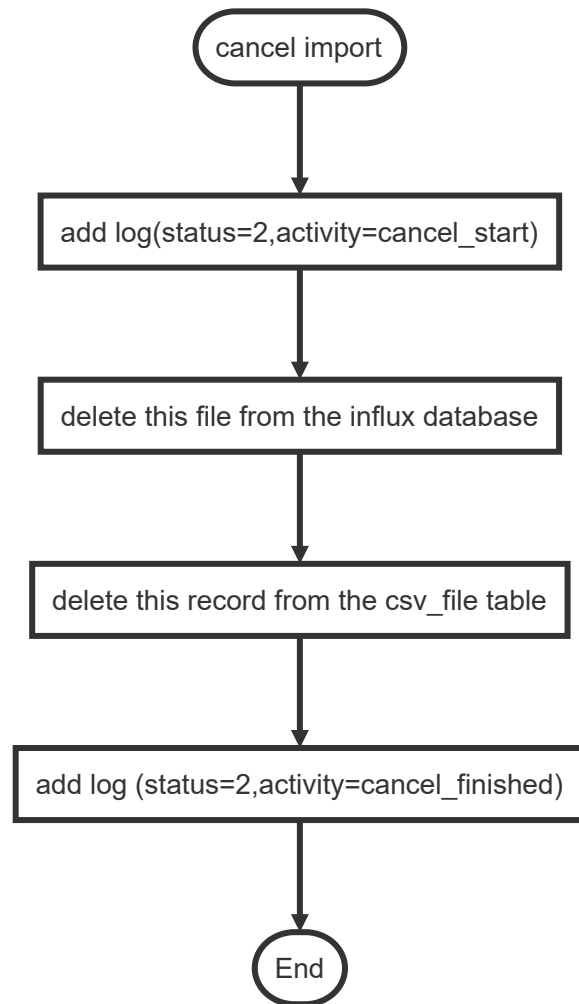
delete a csv file:



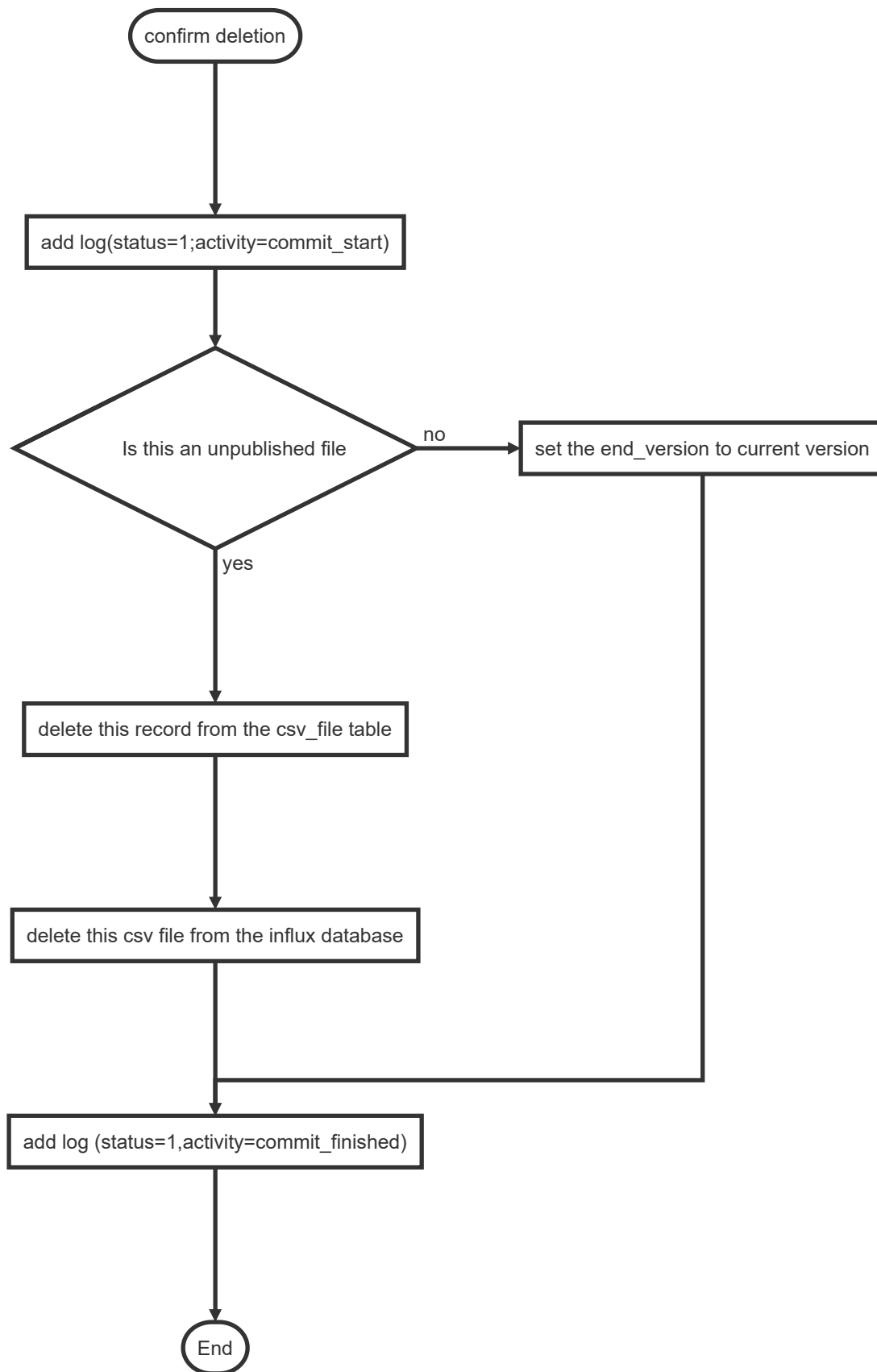
confirm an import:



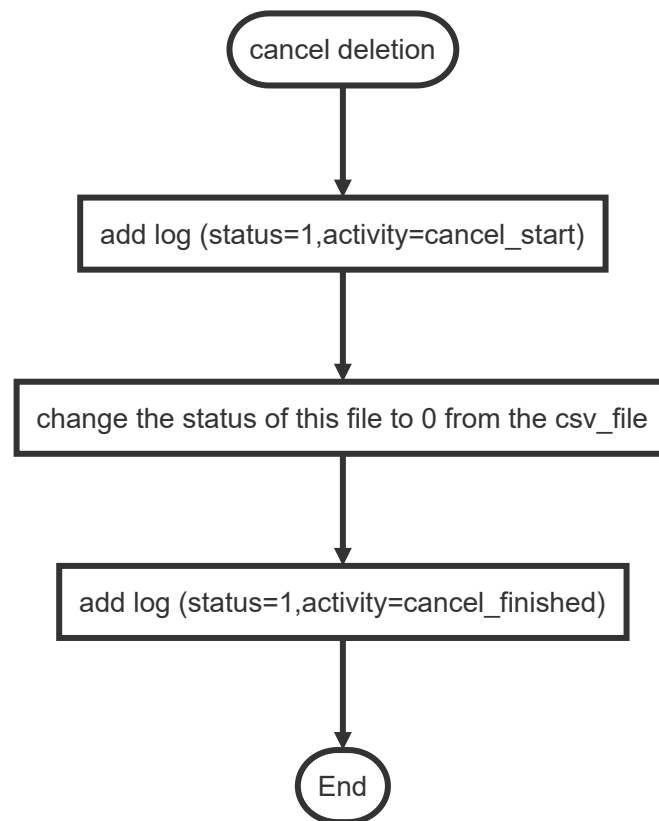
cancel an import:



confirm a deletion:



cancel a deletion:



Version control

In order to make sure that users can access the same data from the database, the version control system is developed. Users can select a different version of the database and get different output. Even if the database is upgraded, users can still get the same data from the old version.

Design

There are three different kinds of data in the database, the EEG data which is imported through .csv files, the patient data which includes basic information about each patient and the medication data which includes medication records of each patient. In order to achieve the version control system. New tags are added into different tables.

For EEG data, "Start_Version" and "End_version" are added. When a new version is published, the current version number would be added to every EEG record whose Start_Version is empty. When a record before the current version is deleted, the End_Version of that record would be set as the current version number. When a record of the current version is deleted, it would be deleted from both MySQL and Influx.

For example, after version 1 is published, all EEG data in the database should have a Start_Version as 1 and an End_Version as $+\infty$ ($+\infty$ is the default value). Now, some new EEG data are imported, the administrator can do the data cleaning job to delete some wrong data. If a record from version 1 is detected to be wrong(Should avoid this situation, try to clean the data before publish), the administrator can also delete it and the End_Version of this record would be set as 2 which means this record would not be used since version 2. When the data cleaning job

finished, version 2 could be published and all data without Start_Version would be set as version 2. If a user wants to use version 1 database now, the system would select all the data with Start_Version ≤ 1 and End_Version > 1 .

In summary, if the user wants to see the i th version of the database, all data with Start_Version $\leq i$ and End_Version $> i$ would be shown.

For patient data, since old patient records would not be deleted, only one tag called version is added. If the user wants to check the i th version of the database, all patient records with version $\leq i$ would be shown.

For medication records, they are the same as patient data, one tag called version is enough.

Export function

Since it is hard to add a new column in influxDB, filenames in MySQL are used to implement version control in influxDB. For every export job, filenames which can be used for this version would be gathered. When generating the influxDB query, filenames would be added to the where clause. In this way, users would get different export from different versions of the database.

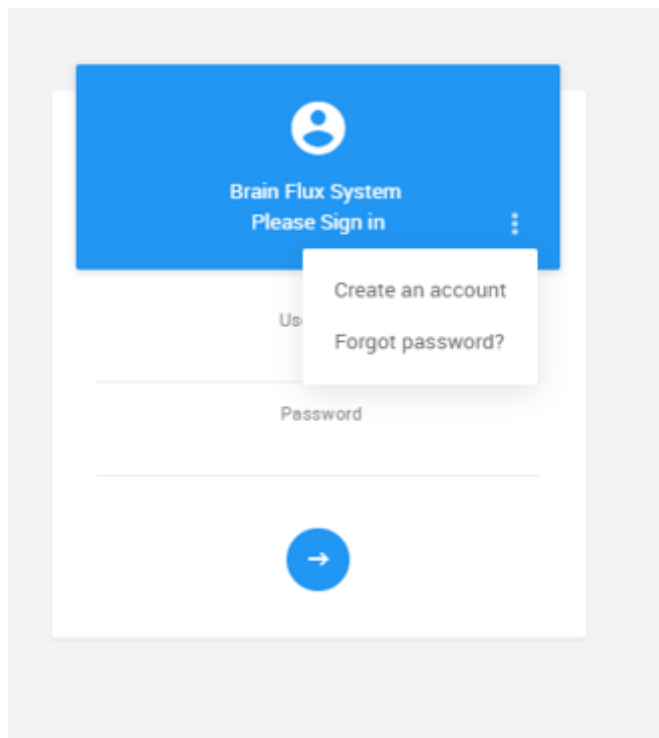
Import function

Since the filename is used as a key to finding data in influxDB, it is important to make sure that the same file cannot have the same name in different versions of the database. For example, file "PUH-2010-001_ar.csv" is detected to be wrong after version 1 published. The administrator can delete it and reimport a correct one for future versions. In this situation, the system would add a version number before this filename to distinguish the file for version 1 and version 2. The name of this file in version 2 would be "V2_PUH-2010-001_ar.csv". On the other hand, if a file is detected to be not complete after published, the administrator cannot import the rest part with the same filename directly, the system would not allow this. The administrator should delete the file and then reimport the complete file.

User management

In the Brainflux system, there are two kinds of users: users who use data from the existing database for the further research, and administrator who can manipulate files in the database (e.g deleting files, writing a comment) and manage other users (e.g resetting passwords, changing access authorities). To distinguish these two users, we implement a user management system using spring boot security.

Register and log in



This is an interface for users to login using user name and password. After creating a new account, the default authority only allows users to export from the static database, to upgrade the authority, users should contact administrators.

when users forget their password, contact administrators and they will reset your password to the default "123456"

Profile page

Users can click on the "view profile" button on the right top drop-down menu and jump to the user profile page. Users can edit basic information, select database version or change password in this page.

User management page

Only an administrator can access this page and operate for the users, including changing user information, resetting the password, and disabling/enabling users.

Crash recovery

Data export and import is a long process in this system. When the system crash happens, the current processes would be interrupted, how to recover them becomes a problem.

Import

When the system crashes during the import process, the following 5 steps can help recover the database and reimport the unfinished files.

step1: Restart the website. It can be restarted by running the deploy script, which is called "restart.sh" in the BrainFlux folder.

step2: Check the unfinished files in MySQL database. Open MySQL workbench and type

```
SELECT filename FROM upmc.import_progress where status<>"STATUS_FINISHED" and status <>"STATUS_FAIL";
```

in one SQL file and run it. The result would be the path of every unfinished files. Copy this and it will be used later.

step3: Delete the ongoing process in MySQL import_process table by typing the following command in one MySQL SQL file and run it.

```
Delete * FROM upmc.import_progress where status<>"STATUS_FINISHED" and status <>"STATUS_FAIL";
```

If error 1175 occurred, type

```
SET SQL_SAFE_UPDATES = 0;
```

and run it then repeat step3.

step4: go to the path copied in step2, delete all file locks(.lock). The file locks are hidden files, change the system to show them and delete them.

step5: After former 4 steps, there should be no ongoing process in the import activity page. Open the import page again and type the file path copied in step2 without last filename and click the import button, if there are more than one directories, do them one by one. Select all function can be used and there could be some files fail to be imported. It is fine since those files were imported successfully before the crash.

Auto-import

There are several possible reasons why auto-import failed, for example, network disconnect, computer breakdown, etc. Depending on which step it failed, the file would be indifferent directories:

- (1) if the EEG files still in the pre-defined folder, which means the .csv files are generated failed, the easy way to recover is to clean all the files in the directory where stores the output of PSCLI, the system will retry the import in the next day.
- (2) if the .csv files still in the target folder where receives the files from windows computer and there is no associate lock, that means the system may be crashed during sending the CSV files, you need to delete the CSV file in the receiving folder on the server, and make sure the original EEG files are in the windows pre-defined scan folder.
- (3) if there are file locks in the receiving folder, it means the ordinary import function is failed. you need to follow the instruction in the last section to deal with this type of error.

Export

If the system crashes during export, it would not resume the export process when system restart, therefore, there may be export jobs in the "Export / Export Jobs" that show blue "checking" button in the result column with no changes of progress column.

In this situation, users should delete the ongoing job files in the '/ tsdb / output' folder and their zipped files in the '/BrainFlux/archive' folder, delete the corresponding job record in the 'job' table by typing the following command in one MySQL SQL file and run it.

```
delete from export where finished !=1;
```

And then start a new job with the same export setting to redo the process.

Data maintenance

In order to make sure that all data in both databases are safe and to save some storage space, data backup and data clean are necessary.

Database backup

Log cleaning

Since there are a considerable amount of data, the size of the database increase very fast, there are several tables that you can clean once you make sure the data is clean and stable.

(1) 'import_log': this table grows rapidly since the new record is added every 15 seconds during the import process, and most of them are useless. Running the following command in MySQL can only keep the latest 100 records in this table.

```
delete from import_log where id not in (SELECT id FROM upmc.import_log order by timestamp desc limit 100);
```

(2) 'csv_log': this table contains all the operation commit to the files in the database, such as Import, deletion, and their corresponding confirmations. So only clean the outdated records when the stable version is published. Running the following command in MySQL can only keep the latest 100 records in this table.

```
delete from csv_log where id not in (SELECT id FROM upmc.csv_log order by timestamp desc limit 100);
```

Export job cleaning

The export files take up large space in the service computer, so it's important to delete outdated export files. The following steps will help you with deleting useless export files from the back-end:

1. Delete the job from the 'export' table using their job id, which appears on the 'export/export jobs' page by running the following SQL command in MySQL.

```
delete from export where id = (the job id you want to delete);
```

2. find the corresponding export files in "tsdb / output" folder, which contains the alias of this job.
3. the zip file for this job is stored in 'BrainFlux / archive ', delete them to make it consistent.

System maintenance

There are several common problems may occur while maintaining the system:

Prerequisite for the development

Since our system is built based on Spring Boot, you need to set up the environment for development, Here are the components you need for this project:

JAVA

You can find the package and the instruction you need below: https://www3.ntu.edu.sg/home/ehchua/programming/howto/JDK_Howto.html

Integrated Development Environment(IDE)

The typical compiler we used is IntelliJ IDEA, you can find the download link below: <https://www.jetbrains.com/idea/download/>

Notice that we use Mybatis to retrieve and update the information of SQL database, so please make sure your editor can use mybatis generator.

MySQL database

We use MySQL to store the metadata of the patients' files, so you need to install the SQL database on your machine to test the functions such as user management, files validation, etc.

The schema of the database is saved as a SQL file that shared with you, run this file to construct the structure of the database.

Also, you need to modify the configuration in the corresponding files, to be specific, you need to change the username and password of in the 'application.properties' that shared with you.

Influx database

The influxDB are used for storing the actual EEG data. Each patient is stored in one measurement, and tags are used to distinguish data from different files, and different processing method. Therefore, the influx database must be installed before using the export function.

Project codes

The codes for this project can be cloned from GitHub(' <https://github.com/tsdb-project/influx-test.git> '). Before running the project, please contact us asking for two more files that are not uploaded to the GitHub.

Deploy the project on the server-side

In the folder "BrainFlux", there is a file named "update_restart.sh", which is used for polling codes from Github and reboot our project. Run this file with the following command in the terminal:

```
sh update_restart.sh
```

The system will update codes and restart in about 30 seconds.

Restart the influxDB on the server-side

If the influx database crashes, all the actions involving export from EEG data will cause errors. The following code is used for restarting the influxDB, it would take a few minutes.

```
influx -config /Volumes/INFLUX_RAID/config/load.ini
```

You should modify the directory if it's changed.

Exceptions handling

If the terminal returns error: "The port has been used", you should change the bind-address in the load.ini file.

Add new administrator to the system

The new administrator should first register as a new user in the system login page, the default user's role will be "ROLE_USER", and then ask one of the existing administrators to change the role in the user management page.