Seshat

Release 0.1

Apr 27, 2023

Administrator's Guide:

The Seshat annotation manager is a simple system for the automated management of annotation campaigns of speech data. It enables a campaign manager to assign annotation tasks to a pool of annotators, and ensure that annotations all comply with a predefined annotation Schema, easily created through Seshat's campaign creation interface.

- If you're willing to learn how to install Seshat, proceed to *Installing Seshat*
- If you want to add annotations corpora to an already-installed instance of Seshat, or use some of Seshat's advanced features proceed to *Administering A Seshat Instance*.
- If you're a campaign manager and wish to learn how to create annotation campaigns and manage them, proceed to the *Campaign Creation Tutorial*.

CHAPTER 1

Installing Seshat

We've made the installation of Seshat as simple as possible.

- If you don't have access to a Linux server or have very sparse knowledge of how to set up a web server and configure it, you should follow the *Docker Install (Easy way)* procedure. This also goes for more seasoned linux users that don't want to go through the hassle of setting up each parts of the server and client manually.
- If you like to have a finer control of your system and/or don't want to deal with Docker, follow the *Manual Install*

These two tutorial will help you Seshat on your machine, it will thus be only accessible **to your or users on your local network**. If you want Seshat to be reachable from the web, follow *Setting up Seshat to be accessible from the web* after installing.

1.1 Docker Install (Easy way)

This type of install should work on any machine, regardless of the OS (Windows, Mac, Linux, although Unix systems are prefered), as long as you have the following requirements:

- A working install of Docker (we know it works with Docker 19.03 and later, it might work with earlier versions). If you don't have Docker installed:
  - On **MacOS** or **Windows**, follow the install instructions for "Desktop".
  - On **Linux** follow the install instructions for "Sever" (pick the right distibution). Pick the Community Edition (CE) install.

    After installing, you should make sure your current user has the right to run Docker images (by running `docker run hello-word` for instance). If that isn't the case, follow these instructions.
- If you installed docker on Linux (namely, Ubuntu) make sure that you have docker-compose installed as well.
- You should also also have Git installed on your system, to retrieve Seshat's sources. If you don't, follow these intructions.

Once all these prerequisites are all installed, open a terminal, and retrieve the full Seshat sources

```
git clone --recurse-submodules git@github.com:bootphon/seshat.git
```

Then, change the current directory to seshat's previously cloned directory

```
cd seshat/
```

Once you're in that folder, edit (using any text editor) the `docker-compose.yml` file

```
version: "3"

services:
  mongo:
    image: mongo:latest
    tty: true
    stdin_open: true
    container_name: seshat_mongo_db
    ports:
      - "127.0.0.1:27017:27017"
    volumes:
      - ./mongodata:/data/db
    restart: always

  server:
    build: .
    image: 'seshat/server:prod'
    container_name: seshat
    ports:
      - "5000:80"
      - "4200:4200"
    restart: always
    depends_on:
      - mongo
    volumes:
      # set the corpora folder mounting point. You can set it to something different.␣
↪For instance, if you
      # corpora folder is /home/mark/data/my_corpora, this line will be
      # - /home/mark/data/my_corpora:/app/corpora
      # DO NOT change the second part of this line (the ":/app/corpora")
      # DO NOT add another mounting point other than this one for the corpora folder
      - ./corpora:/app/corpora
    environment:
      - FLASK_CONFIG=docker
      - ADMIN_USERNAME=manager_username
      - ADMIN_PASSWORD="your_admin_password"
      - ADMIN_EMAIL="test@test.com"
      - ADMIN_FIRST_NAME="Perceval"
      - ADMIN_LAST_NAME="Le Gaulois"
```

Edit all the parameters defining the first admin's accounts properties:

- For the username `ADMIN_USERNAME` use alphanumerical characters, no spaces
- The password `ADMIN_PASSWORD` should be at least 8 characters long
- The email defined in `ADMIN_EMAIL` should be a valid email address. If you don't want to put any, just leave the default one.
- Regarding the `FIRST_NAME` and `LAST_NAME` variables, any string will do

Note: It's important that you set your password to something secure. **Do not** ignore this step and leave the example password preset.

Now, copy one or more of your corpus's folder to the `corpora/` folder that is already present. Do it either using your OS's file browser, or by running the following command

```
cp -r /path/to/your/corpus/folder corpora/
```

Once that is done, still in the `seshat/` folder, tell docker to build and run your local image. The build part might take some time.

```
docker-compose build
docker-compose up
```

This will automatically download the required Docker images, install them and setup Seshat and its database in a Docker environment. Once you see the message the following messages displayed in your terminal, it means it's finished.

Test your access to the now active application by reaching the address http://localhost:4200 in your browser. Check that you can log in with the username and password you specified in the configuration file. You can now add audio corpora to Seshat, so you can start your first annotation campaign. Check that your corpora is in line with *Seshat Audio Corpus Management*, and just drop it in the `corpora` folder (still in Seshat's current folder), or any other corpora folder that you set up during the configuration.

Once that is done, you can start using Seshat, or learn how to use it in the *Campaign Creation Tutorial*.

Note: You can cut the running docker images with Ctrl-C. If you want them to run in the background, run `docker-compose up -d` (the `-d` stands for daemon).

1.2 Manual Install

This type of install is advised if you're good enough as a Linux system administrator, and don't want to have to deal with Docker. It's considerably easier to tweak and customize the install with this kind of installation.

This install is for an Ubuntu 18.04 environment (it could potentially work on other environments distributions or MacOS but has not been thoroughly tested). We here assume that your system has the following requirements:

- A working python 3.6>= (default Python version on Ubuntu 18.04)
- A working nodeJS 12.3>= install
- A mongoDB 3.6>= install (it could work with earlier versions)
- An ffmpeg install

Moreover, the advised web server to act as a web frontend is **Nginx**.

1.2.1 Preliminaries

First of all, clone the full seshat project and `cd` into it

```
git clone --recurse-submodules git@github.com:bootphon/seshat.git
cd seshat/
```

1.2.2 Setting up the API (server)

First, `cd` into the server's directory, set up a python virtual environment, and install the seshat package:

```
cd server/
python3 -m venv venv/
. venv/bin/activate
python setup.py install
```

You should be able to run the server now, by running:

```
python app.py
```

Check that it is running by reaching the address http://localhost:5000/doc/swagger. You should see a page displaying a description of Seshat's RESTful API.

1.2.3 Setting up the client

Now go to the client's directory (assuming you're still in the server's directory). Then, install the npm dependencies and compile the client's Angular Typescript code into JavaScript:

```
cd ../client/
npm install
npm run build -- --prod true --outputPath=../server/static/
```

Now go back to the server's directory, and run `python app.py` again. If you reach the address http://localhost:5000/, your should get Seshat's login page.

1.2.4 Creating a manager's account

Still in the server's virtual environment, run the command (whilst replacing the example values with your own)

```
add-manager manager_username manager_password manager@email.com --first_name Manager -
↪-last_name Mister
```

If you go back to the http://localhost:5000/ page, you should now be able to login with the supplied username and password. At this point, your installation of Seshat is fully usable, albeit not very practical.

You can drop corpus files in the default `server/corpora/` folder for Seshat to use for annotation. Be sure to comply with the *Seshat Audio Corpus Management*.

1.2.5 Having Seshat as a Daemon

Having to launch Seshat by hand each time you use it isn't very practical. If you're willing to have Seshat an automatic launch of Seshat when your machine (or server) boots, follow these steps to have Seshat "daemonized" using SystemD. Note that if you wish to have seshat served by nginx, it's one of the required steps, thus, this isn't wasted time in any way.

First, make sure you have no already running instance of Seshat (as it will conflict with the deamonized one if their ports are the same).

Edit the `seshat.service` found in the `install/` folder of the clone repository, configuring it with your own paths and your own username:

```
[Unit]
Description=uWSGI instance serving the seshat website
After=network.target

[Service]
User=your_user
Group=www-data
WorkingDirectory=/path/to/seshat/server/
Environment="PATH=/path/to/seshat/server/venv/bin:/usr/bin"
ExecStart=/path/to/seshat/server/venv/bin/uwsgi --ini wsgi/seshat_http.ini

[Install]
WantedBy=multi-user.target
```

Then, copy that file to systemd's services directory, and start the deamon:

```
sudo cp install/seshat.service /etc/systemd/system/seshat.service
sudo systemctl start seshat.service
sudo systemctl status seshat.service
```

The last command should say that seshat's service is active. Seshat should now be running on the same address as usual, http://localhost:5000/.

If you wish to have Seshat running on boot, run:

```
sudo systemctl enable seshat.service
```

If you wish to serve Seshat to the web, proceed to *Setting up Seshat to be accessible from the web* .

CHAPTER 2

Seshat Audio Corpus Management

2.1 Supported Corpus Formats

Seshat supports two types of audio corpora:

- The *File Based Corpus* (audio files in a folder hierarchy)
- The *CSV based Corpus* summaries

2.1.1 File Based Corpus

This kind of corpus is just a file hierarchy with audio files in it. Structure doesn't matter, Seshat will just look (*recursively*) for all files matching an audio extension (*.wav*, *.ogg*, *.flac*, *.mp3*), and consider them to be corpus files. You just have to "drop" that audio folder in the folder you defined as *corpora*.

It's however advised to have a hierarchical structure that describes best your data, as the path to the file will be used as its name when assigning task. Consider for instance the following two small yet **valid** corpora:

One of people saying things:

```
corpus_people/
├── female
│   └── mary
│       ├── 001.wav
│       ├── 002.wav
│       └── 003.wav
└── male
    ├── henry
    │   ├── tis_but_stratch.wav
    │   └── what_is_your_quest.wav
    └── john
        ├── 001_hello.wav
        ├── 001_water.wav
        └── 003.wav
```

Then, let's also consider a corpus of bird songs:

```
corpus_birds/
  ├── unknown_1.wav
  ├── unknown_2.mp3
  ├── summary.docx
  ├── blackbird
  │   ├── 001.wav
  │   ├── 002.ogg
  │   ├── 003.wav
  │   └── 004.wav
  ├── nightingale
  │   ├── 001.flac
  │   ├── 002.wav
  │   └── 003.wav
  └── northern_cardinal
      ├── 001.mp3
      └── 002.wav
```

As you can see, there are no constraints on the file structure, the number of files, or the naming of these files (however, for your own sake, please keep some consistency in your naming conventions). You can also notice that there is a `docx` file thrown in there, that Seshat will just ignore.

The corpus's root folder will be used as the corpus's name by seshat, e.g. here, it's corpus_people and corpus_bird.

2.1.2 CSV based Corpus

If you don't want to have the actual files from your corpus on the same machine as the Seshat website (for security reasons for instance), you can still provide seshat with a CSV file that summarizes your corpus. It only requires **two** columns: one for the file ID (idealy, human readable and descriptive, not a md5 hash), and its duration in seconds.

For instance, this is a valid csv `people_talking.csv`:

```
filename, duration
john_001.wav, 13.432
john_002.wav, 10.13
john_003.wav, 42.045
mary_001.wav, 9.454
mary_002.wav, 2.43
```

The CSV file's name will be used as the corpus name by seshat, e.g., here it'll be people_talking.

Note: For CSV-based corpora, Seshat won't (quite naturally) be able to serve the audio files to the annotator, as they aren't available (even if you use a valid path as the file name).

2.2 Importing Your Corpora to Seshat

Importing a corpus to Seshat is a easy as dropping (or copying) a your folder or CSV file to Seshat's *corpora/* folder. You should go to *Import a Corpus (Docker Install)* or *Import a Corpus (Manual Install)* depending on your install to learn how to do that.

CHAPTER 3

Administering A Seshat Instance

3.1 Managing Seshat

Here, we'll teach you how to manage Seshat as the administrator, **not as the campaign manager**, on a **Docker** or **Manual** install of seshat. .. TODO : ref the manual and docker installs

3.1.1 Adding Audio Corpora

On a Docker Install

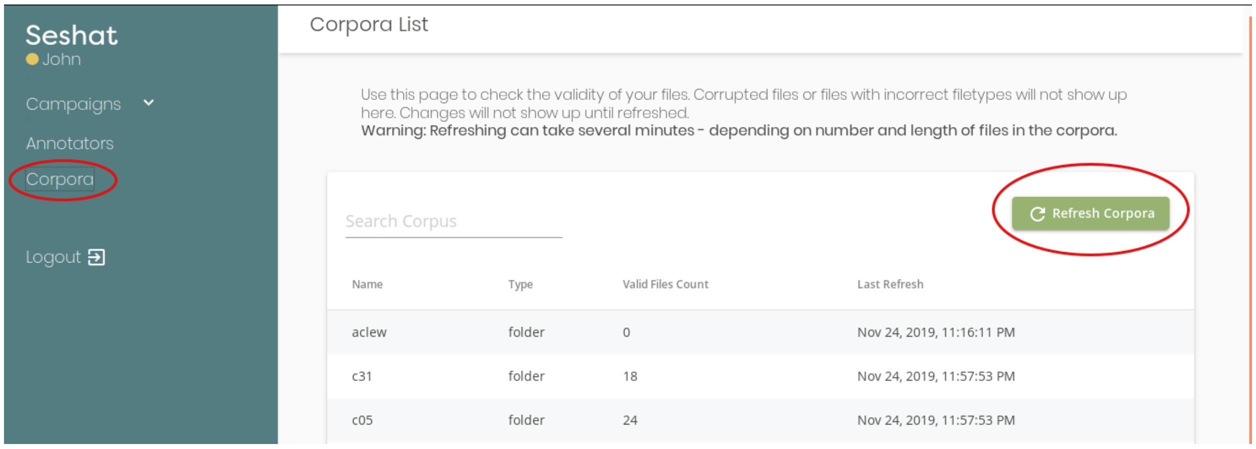
Seshat's corpora folder is where you should drop your audio corpora. It's set up in your `docker-compose.yml` file. If you left it unchanged, it should be the `corpora/` folder in seshat's clone repository. Else, it's the left side of the line:

```
volumes:
  # here it's '~/datasets/' for instance
  - ~/datasets/:/app/corpora
```

You should then just drop **corpus's folder (or CSV file) inside the special corpora folder**, not your corpus's content. For instance, say you have 3 corporas, corresponding to 3 folders, then your special *corpora/* folder should look like that:

```
corpora/ <--- this is seshat's "magic" folder
├── some_corpus_summary.csv
├── corpus_1
│   └── some files...
├── corpus_2
│   └── some other files...
└── corpus_3
    └── some more files...
```

After you've dropped a new corpus, or updated a corpus (by adding files to it) into seshat's *corpora/* folder, you should then reach Seshat's "Corpora" page and refresh either the corpora list or the corpus's content:



Use this page to check the validity of your files. Corrupted files or files with incorrect filetypes will not show up here. Changes will not show up until refreshed.
Warning: Refreshing can take several minutes - depending on number and length of files in the corpora.

Name	Type	Valid Files Count	Last Refresh
aclew	folder	0	Nov 24, 2019, 11:16:11 PM
c31	folder	18	Nov 24, 2019, 11:57:53 PM
c05	folder	24	Nov 24, 2019, 11:57:53 PM

Note: You don't have to restart Seshat's docker instance to add new corpora!

On a Manual Install

TODO

3.1.2 Reaching your Seshat Server Environment

On A Docker Install

To reach into your Seshat docker, run

```
docker exec -it seshat /bin/sh
```

If that doesn't work, list your running docker images with the following command, and check that a seshat docker is indeed running.

```
docker ps
```

On a Manual Install

TODO

3.1.3 Creating Manager Accounts

Let's say you want to add another manager's account with the following properties:

```
First/lastname : Provençal Le Gaulois
Email : provencal.gaulois@gmail.com
Login : provencal
Password : freewales
```

First, *reach into your server's environment*. Then, just run the command:

```
add-manager provencal freewales provencal.gaulois@gmail.com --first_name Provençal --
↪last_name Le Gaulois
```

3.1.4 Setting up Seshat to be accessible from the web

On a Docker Install

TODO

On a Manual Install

TODO

3.2 Advanced Features for Seshat

For these features, it's advised to have some basic knowledge of python, and especially of the Python toolchain (pip, environments and python packages).

3.2.1 Adding custom parsers

If you want to know how to implement your own annotation parser (used to check annotations in a specific Tier), you should visit the template's page.

If you've already implemented a parser, or found one that fits your need and want to use it. Let's now install it. First, *reach into your server's environment*. We can now install your parser package. The install command is dependant on the type of package for your parser, but there usually are only two possibilities:

- Your parser package is in a github repository (most likely): run

  ```
  pip install git+git://github.com/myuser/seshat-parser-myparser
  ```

- Your parser package is hosted on pypi (regular type of install but rare for small custom packages): run:

  ```
  pip install seshat-parser-parsername
  ```

Your parser package should now be installed in seshat's python environment and automatically detected by the platform. You should be able to find its installed parsers in the campaign creation view, when creating tier. However, if you wish to check that your parser-package is effectively detected by seshat, and a that the parsers it makes avaible have a valid implementation, you can run

```
check-parser --list
```

This will show you all detected parser names. You can check an individual parser by running:

```
check-parser --parser ParserName
```

3.2.2 Using the Seshat CLI

As shortly mentioned in the *On a Manual Install*, Seshat includes a CLI (command line interface) that enables its administrator to manage it using bash commands (in effect, you can manage it using bash scripts).

Here is a table listing all the CLI commands that are available, with a short description. If you need more information on each command, *make sure you're inside your server's environment image* and then run a command with the `--help` argument, e.g. : `assign-task --help`.

Command	Description
add-manager	Add an annotation manager to your Seshat instance.
add-annotator	Add an annotator account to your Seshat instance.
delete-annotator	Remove an account from your Seshat instance.
update-password	Update the password of a user (Manager or Annotator).
check-corpus	List and check the content of corpora imported by Seshat.
check-parser	List and check installed parser modules.
assign-task	Assign new tasks to annotators on an annotation campaign.
delete-task	Delete one or several assigned tasks.
list-tasks	List assign tasks for a campaign.
campaign-gamma	Retrive the campaign gamma scores (inter-rater agreement) for a campaign.

3.2.3 Using the RESTful API

In case the Seshat CLI doesn't fit your needs, and you'd like to administer the platform from an actual programming language (i.e., not bash), you can make API calls directly to the Seshat RESTful Server. The Seshat server serves its documentation in three formats (thanks to Flask Smorest):

- OpenAPI 3.0.1 API specification in JSON (at the address `/doc/openapi.json`)
- SwaggerUI description (at the address `/doc/swagger`)
- Redoc description (at the address `/doc/redoc`)

Note: The address is prefixed by your server's address. If you're running (or accessing) Seshat's RESTful server locally, it should be (for the swagger doc, for instance) http://localhost:5000/doc/swagger.

You should probably start by looking at the Swagger/Redoc description, which will provide you with a very user-friendly interface that lists all of the API's endpoints. The server's implementation sanitizes and checks all inputs, so you shouldn't be too much worried about messing up your instance if you sent it some ill-formed request.

Warning: To run any API call, you have to call the login endpoint first, which will return a login token. You should then include that token in *each* of your API calls, either as a GET argument (`?token=mytoken`) or in the HTTP request's `'Auth-Token'` header.

You can either build each request by hand, but we advise that you use some tool that takes advantage of the OpenAPI 3.0.1 specification to communicate with Seshat's RESTful server.

If you require any help on how to use the API to do something, don't hesitate on asking us a question on our Github Page.

CHAPTER 4

Custom Parsers

Custom parsers are a powerful way for you to customize the way you want to check annotations during an annotation campaign. In this page, we'll tell you how they work, and what you should implement in your own parser to integrate it in Seshat.

TODO

CHAPTER 5

Campaign Creation Tutorial

This guide will teach you how to use Seshat as a Campaign Manager (the person in charge of assigning annotation tasks and setting the annotation rules).

Before being able to follow this guide, make sure you have a working install of Seshat, and that you have access to a manager's account.

5.1 Creating the Campaign

First, let's give a bit of context for our example. Let's say that we have a corpus of audio files that are recordings of ADHD subjects that were orally given a series of three oral tasks designed find out more about their cognitive processes.

We want our annotators to :

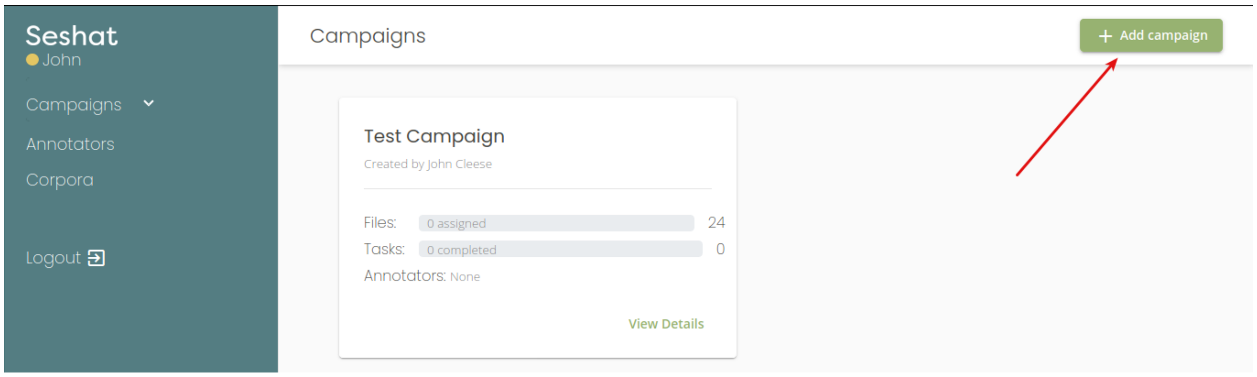
- clearly delimit where each task begins and ends, knowing that each of these are in a contiguous sequence. These tasks are named *MEMORY*, *ATTENTION*, *INVENTION*.
- annotate the transcript of the subjects' speech.
- less importantly, indicate when the interviewer is talking (*INT*), or if there's another person talking (*OTH*),
- indicate if there's some distinguishable noise at that might decrease the recording's quality (*NOI*).

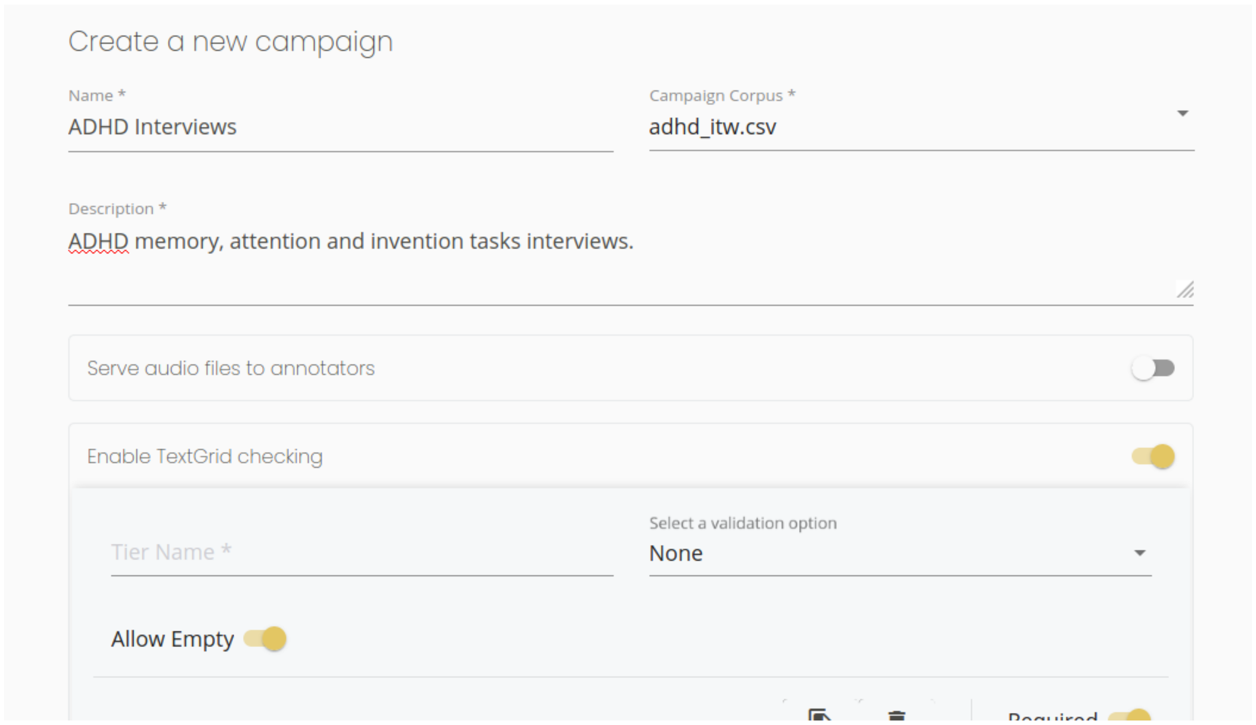
It's now quite clear that we want annotators to produce a 3-tiered annotation for each audio file, with a 4th optional tier in case the interview is noisy. Let's get down to work and create our annotation campaign.

First, reach the **Campaigns** page, and click **Add Campaign**.

Then, give your campaign a name, a description, select the right corpus, and enable the the textgrid-checking option.

Then, let's specify a first tier. This one will contain annotations indicating the current oral task for the subject. These task are contiguous, so no "empty annotations" are allowed in this tier. To ensure that this condition is respected by annotators, deactivate the **Allow Empty** slider. Also, set its **Validation options** to categorical (with categories *MEMORY*, *ATTENTION*, *INVENTION*).




Create a new campaign

Name * ADHD Interviews

Campaign Corpus * adhd_itw.csv

Description * ADHD memory, attention and invention tasks interviews.

Serve audio files to annotators

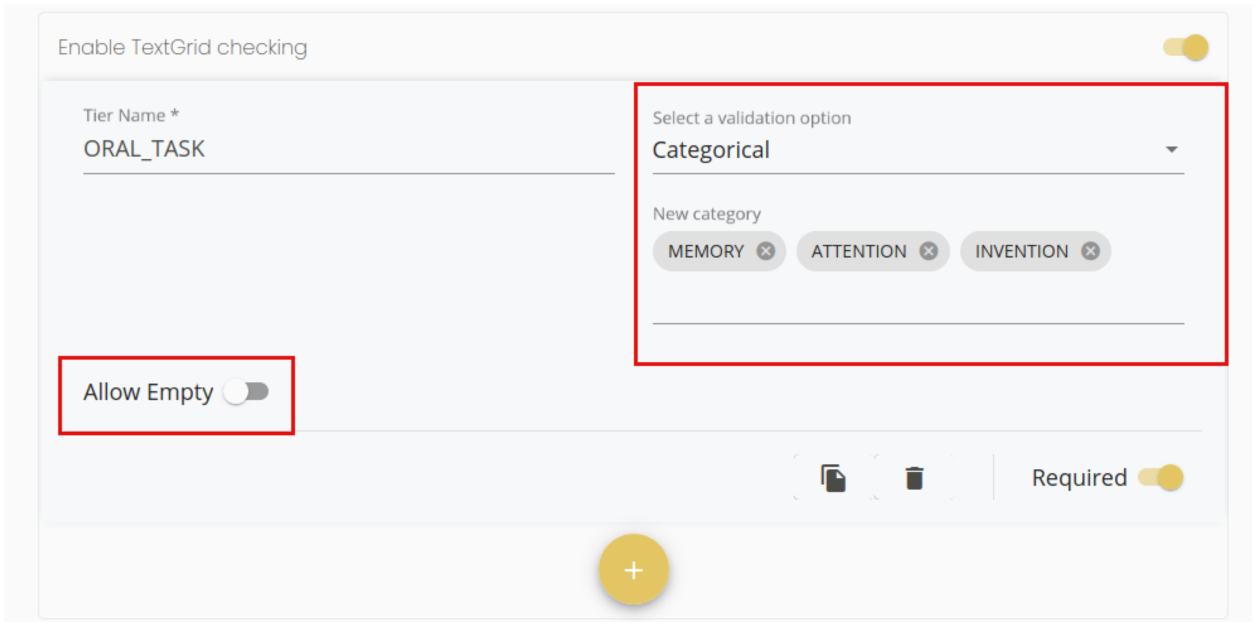
Enable TextGrid checking

Tier Name *

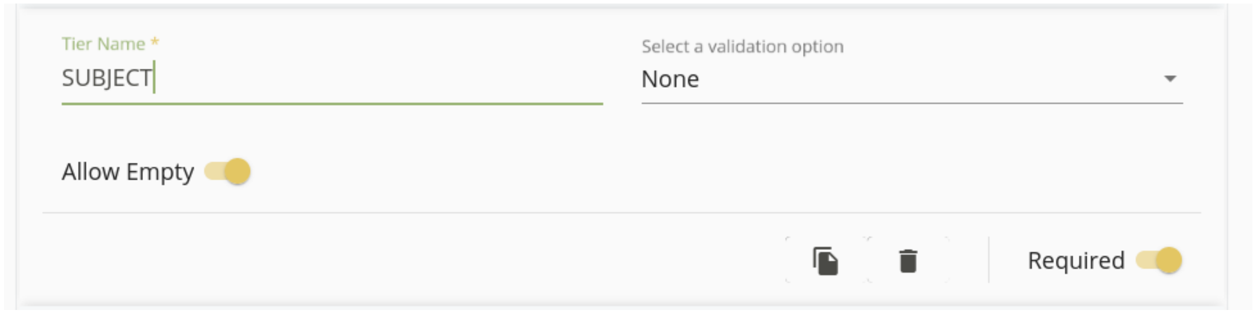
Select a validation option None

Allow Empty

Required



Second, use the "+" button to add another tier. This one, called *SUBJECT*, will be for the subject's speech . Annotations for this tier cannot be checked (it's arbitrary text), thus, leave "Validation Options" to "None".



At last, add the two last tiers. The first one, called *NON-SUBJECT*, will be for *INT* and *OTH* annotations. The other one, called *NOISE*, is for optional *NOI* annotations, and is not always required. Annotators will thus be able to delete this tier if they consider it to be useless. Thus, deactivate the **Required** slider.

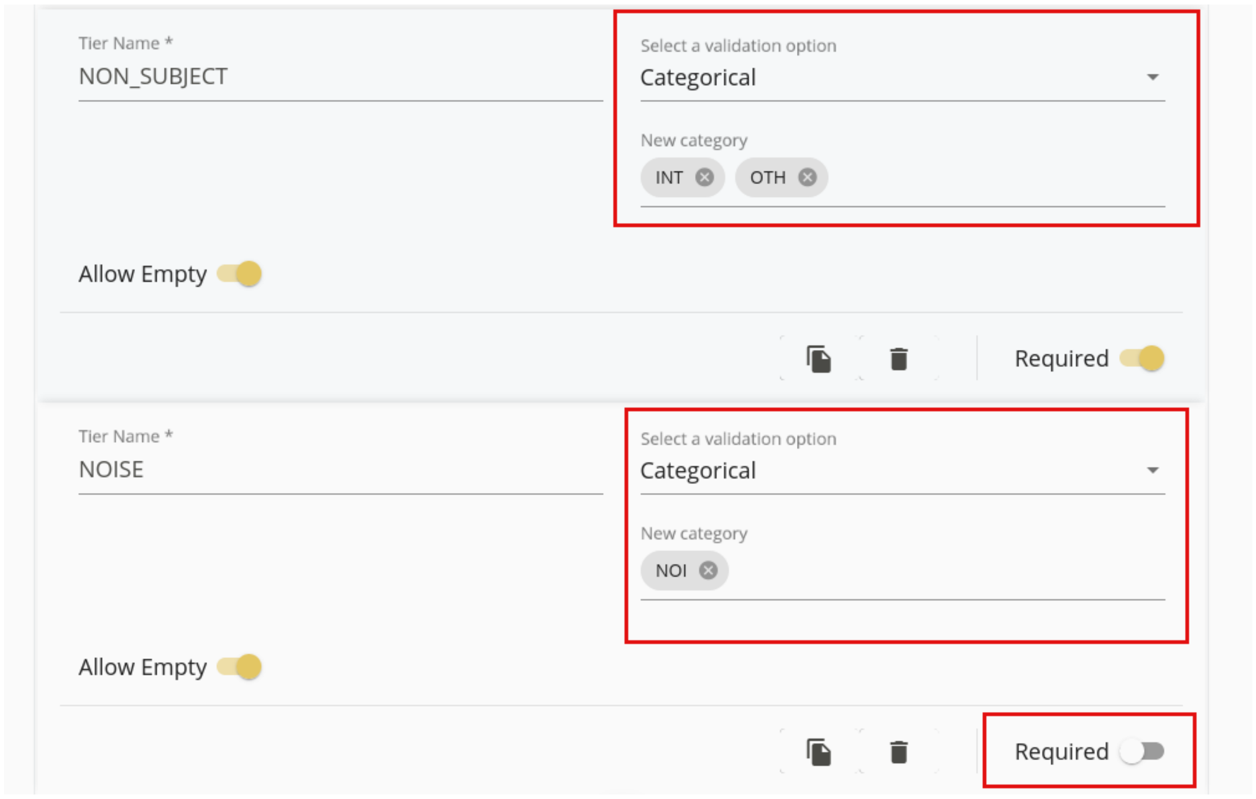
Finally, click **Create Campaign** to finish the campaign's creation. You should be redirected to your campaign's page.

5.2 Creating Annotators Accounts

Before being able to add annotation tasks, we first need to create annotators account for our valiant scribes. In our case, we only have two of them: **Anubis** and **Seth**. Reach the **Annotators** page, then click **Add annotators** to create annotators accounts.

5.3 Assigning Tasks

Now, let's go back to our campaign. Scroll down a bit, and click the **Assign task** button to create new annotations tasks for your two annotators.


Tier Name *
NON_SUBJECT

Select a validation option
Categorical

New category
INT OTH

Allow Empty

Required

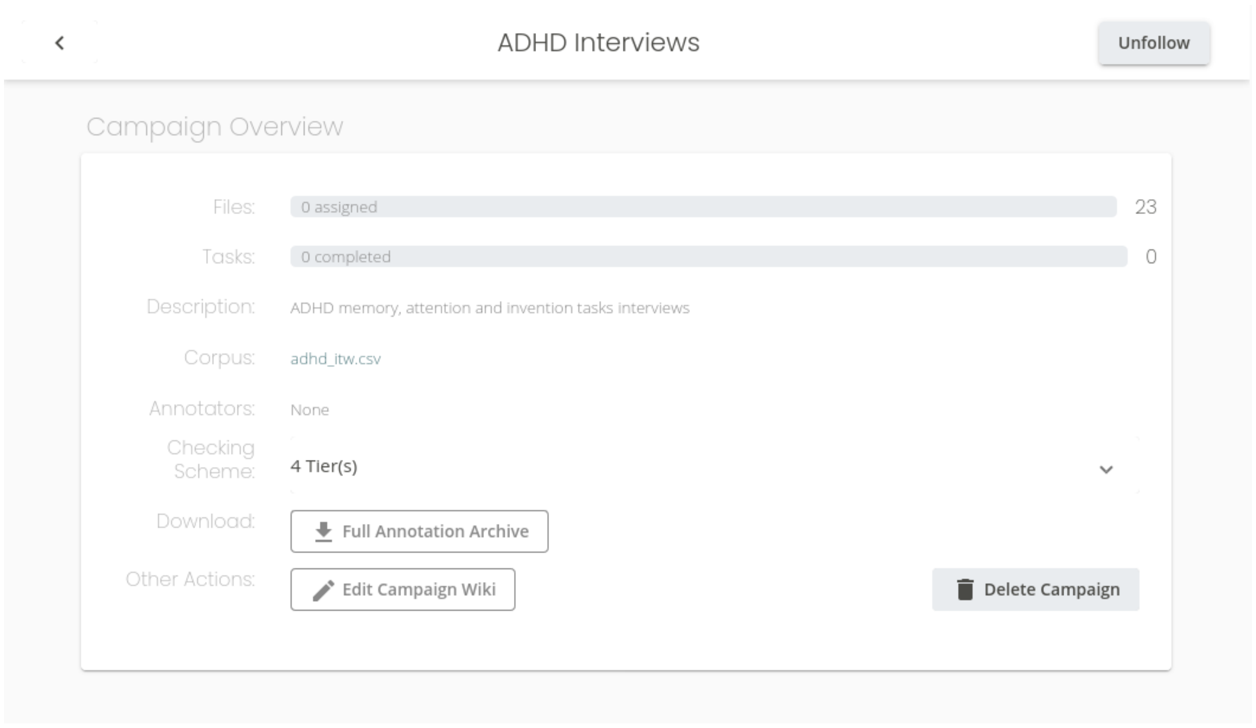
Tier Name *
NOISE

Select a validation option
Categorical

New category
NOI

Allow Empty

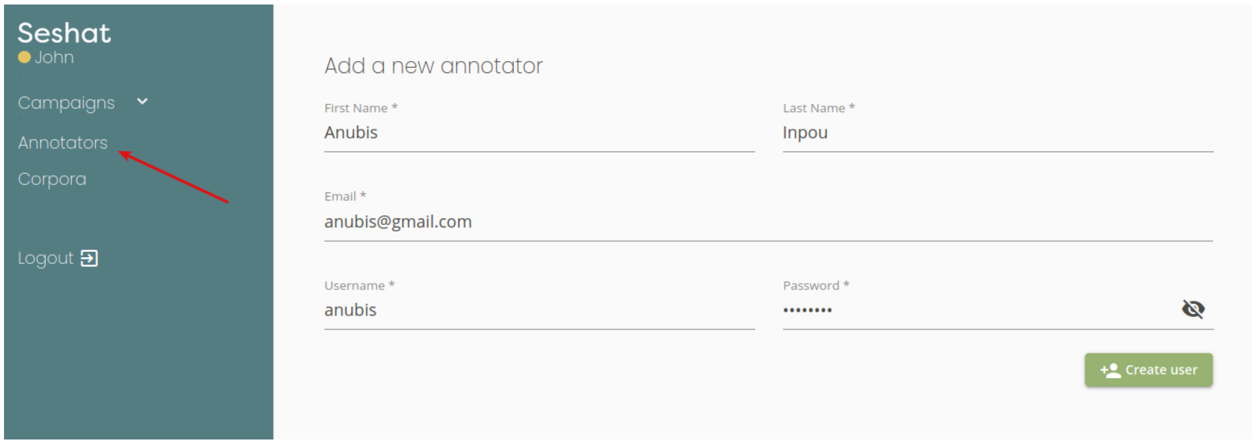
Required


ADHD Interviews

Unfollow

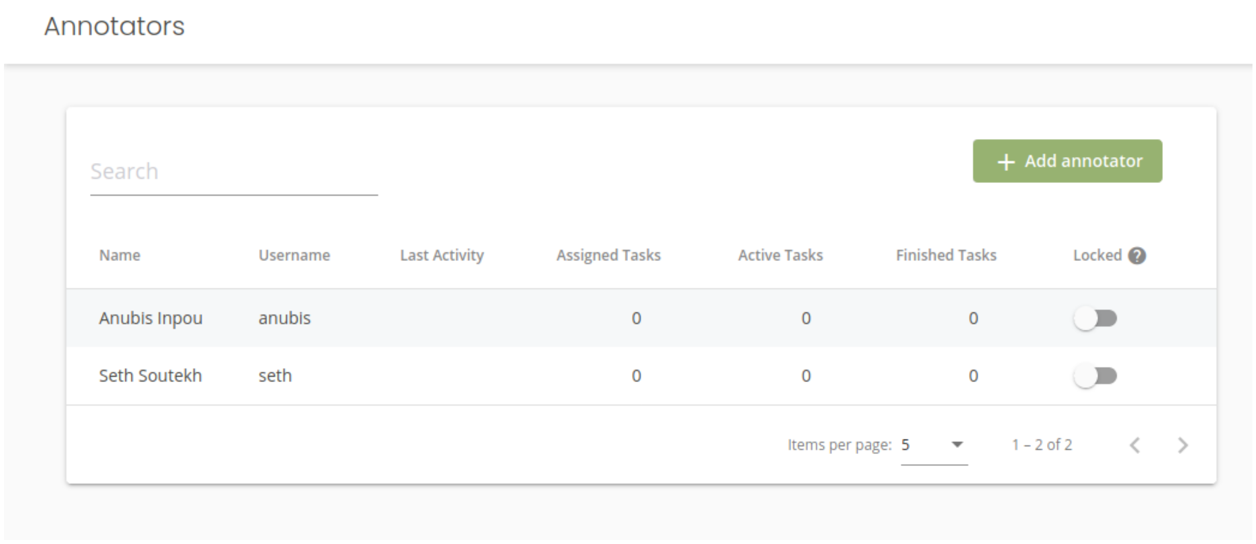
Campaign Overview

Files:	0 assigned	23
Tasks:	0 completed	0
Description:	ADHD memory, attention and invention tasks interviews	
Corpus:	adhd_itw.csv	
Annotators:	None	
Checking Scheme:	4 Tier(s)	
Download:	Full Annotation Archive	
Other Actions:	Edit Campaign Wiki	Delete Campaign



Add a new annotator

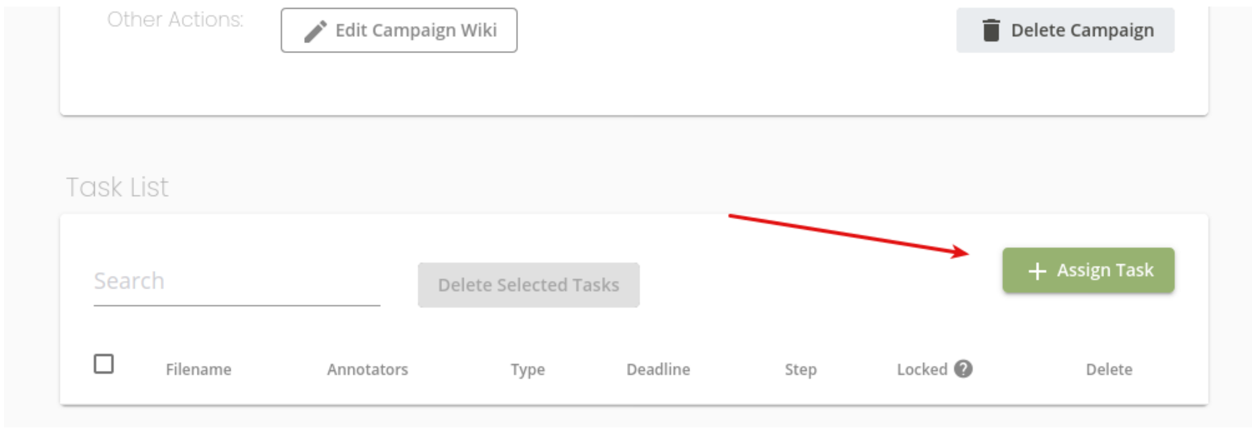
First Name *	Anubis
Last Name *	Inpou
Email *	anubis@gmail.com
Username *	anubis
Password *	••••••••



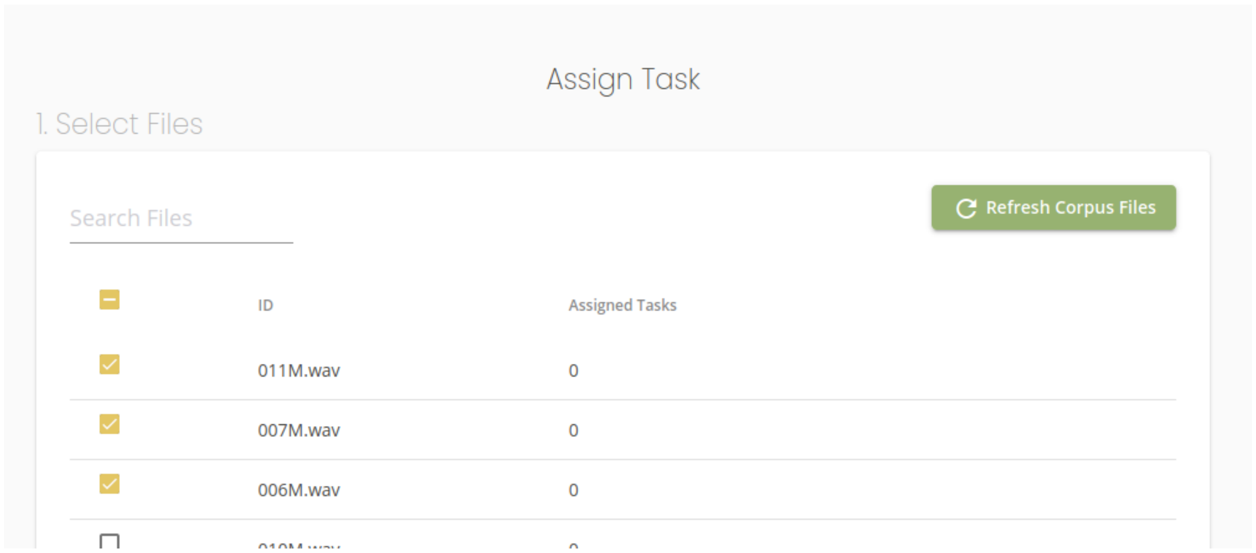
Annotators

Name	Username	Last Activity	Assigned Tasks	Active Tasks	Finished Tasks	Locked
Anubis Inpou	anubis		0	0	0	
Seth Soutekh	seth		0	0	0	

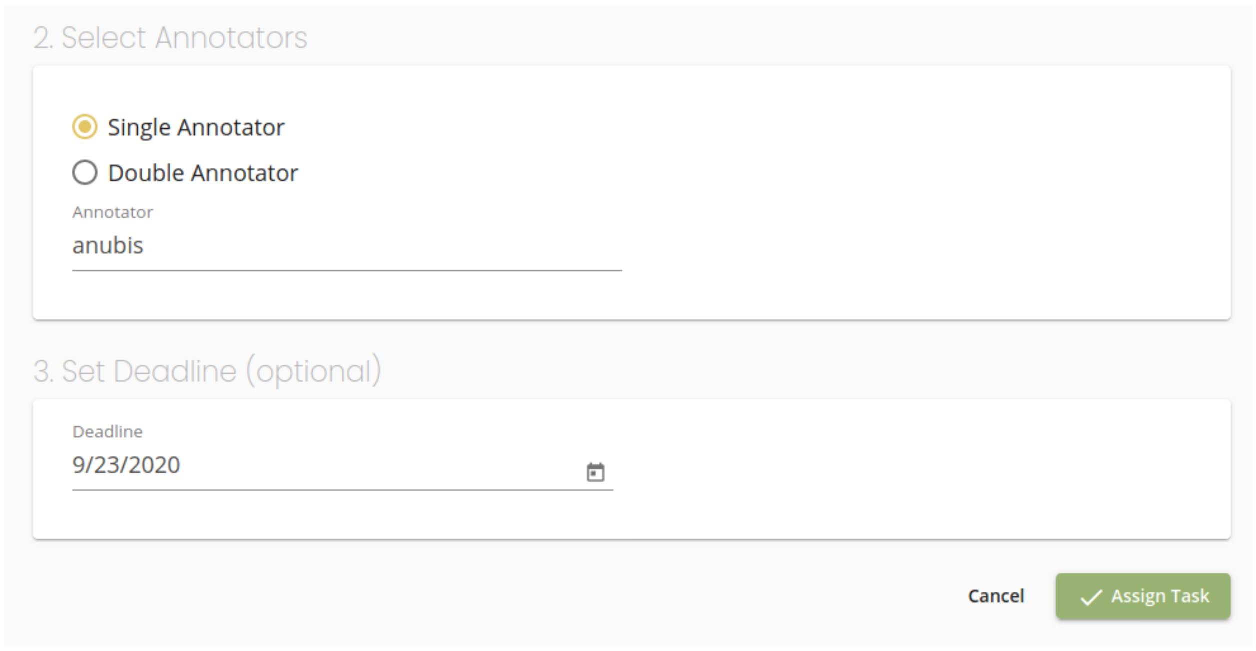
Items per page: 5 1 – 2 of 2



Then, assign 3 annotation tasks to anubis, each naturally corresponding to a different audio file. Select the **Single Annotator** option and fill-in anubis's login. The **Deadline** option can be set as an indicator to help the annotator know what tasks are more urgent, but has no real effect.



	ID	Assigned Tasks
✓	011M.wav	0
✓	007M.wav	0
✓	006M.wav	0



[● Single Annotator] ○ Double Annotator

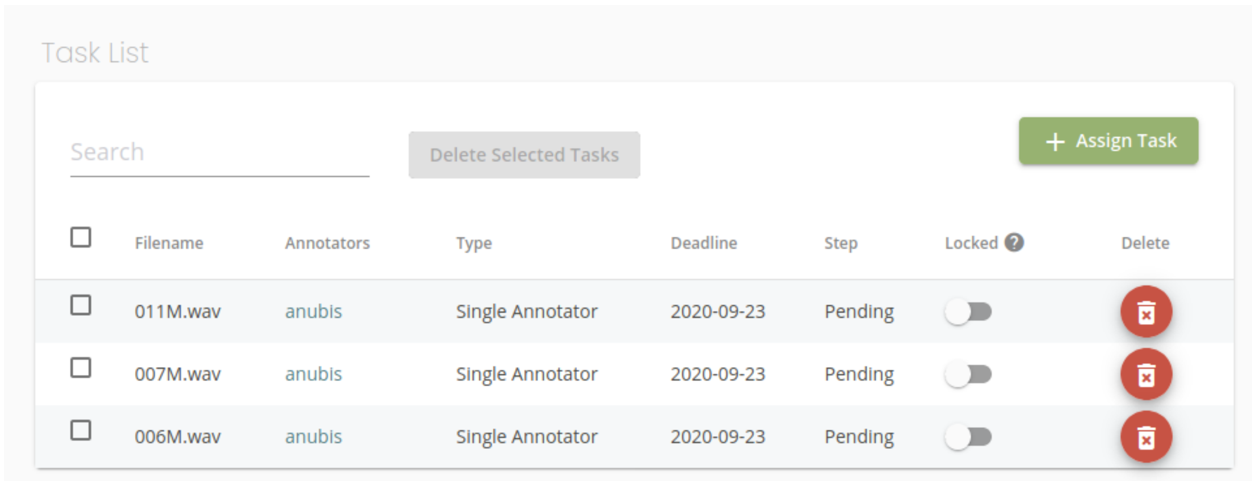
Annotator: anubis

Deadline: 9/23/2020

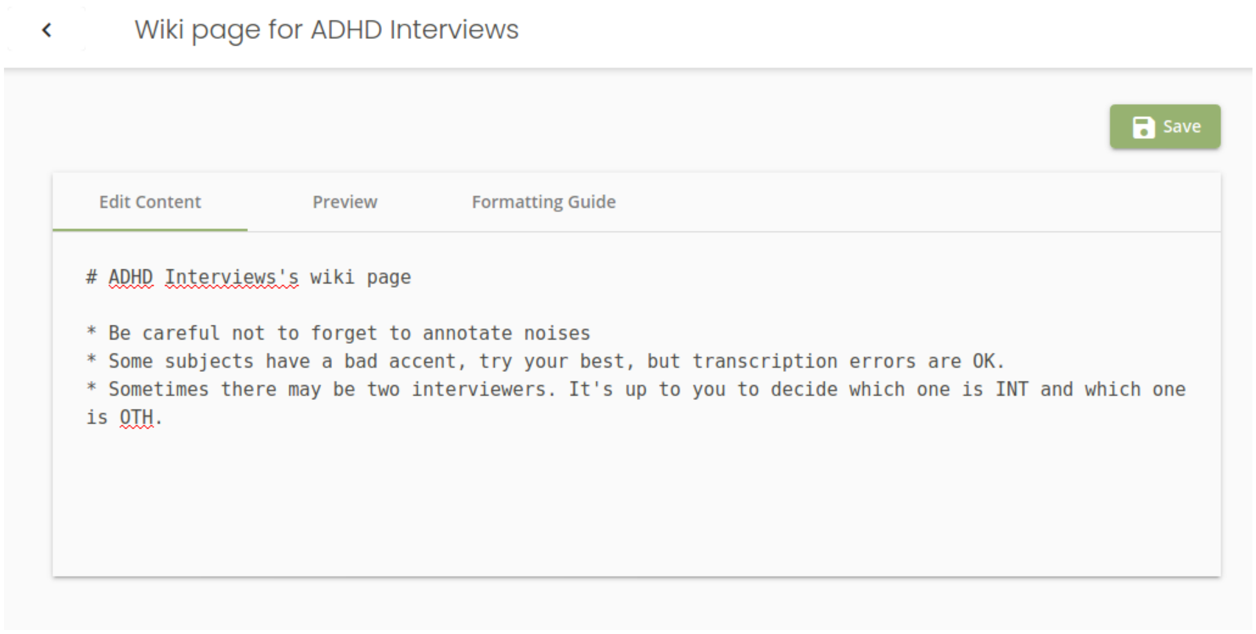
Then, click **Assign Tasks**. This will create 3 tasks for Anubis. Repeat this process for Seth. this will create a number of tasks that can be viewed in the campaign's page.

5.4 Helping your annotators with the Wiki

If you want to give your annotators some extra help on how to annotate the audio file, use the **Edit Campaign Wiki** option. Anything written in the wiki will be displayed in the task's help, on the annotator's own interface.



☐	Filename	Annotators	Type	Deadline	Step	Locked	Delete
☐	011M.wav	anubis	Single Annotator	2020-09-23	Pending		
☐	007M.wav	anubis	Single Annotator	2020-09-23	Pending		
☐	006M.wav	anubis	Single Annotator	2020-09-23	Pending		



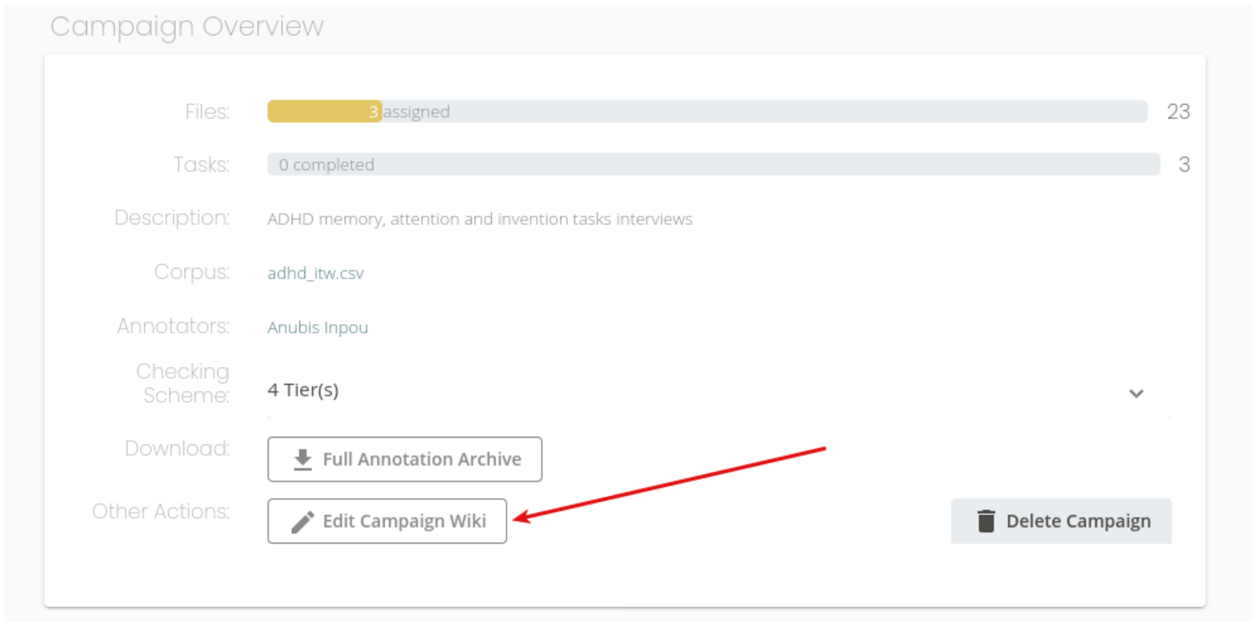
```
# ADHD Interviews's wiki page

* Be careful not to forget to annotate noises
* Some subjects have a bad accent, try your best, but transcription errors are OK.
* Sometimes there may be two interviewers. It's up to you to decide which one is INT and which one
is OTH.
```

5.5 Tracking the Campaign's Progress

# TODO

5.6 Retrieving Annotated Files

At any time during the annotation process, you can download all the annotation files as a nifty ZIP archive by clicking **Full Annotation Archive** on the campaign's page



Campaign Overview

Files:	3 assigned	23
Tasks:	0 completed	3
Description:	ADHD memory, attention and invention tasks interviews	
Corpus:	adhd_itw.csv	
Annotators:	Anubis Inpou	
Checking Scheme:	4 Tier(s)	
Download:	Full Annotation Archive	
Other Actions:	Edit Campaign Wiki	Delete Campaign

CHAPTER 6

The Campaign Manager's Reference

In this page, we define and explain the terms introduced in Seshat. Think of this page as Seshat's rulebook.

6.1 Audio Corpus

A set of audio/speech files that a *Campaign Manager* wants to annotate. It is indicated either by a folder containing sound files, or by a CSV summarizing a set of files. Seshat support the same formats as Praat so far: WAV, Flac and MP3.

For more information on how to format your audio corpora and import them into seshat, take a look at *Seshat Audio Corpus Management*

6.2 Annotation Campaigns

An object that enables the *Campaign Manager* to assign *Annotation Tasks* to the *Annotators*. It references an *Audio Corpus*, and allows the Manager to track the annotation's tasks progress and completion in real time.

6.2.1 Campaign Configuration

When creating a campaign, there are a small number of properties you can configure, which will be explained here. Let's see a screenshot of the campaign creation interface:

Let's go over the different parts of this interface, highlighted by the red boxes:

1. This is where you set the campaign's basic properties:
    - Its name (from which a *slug* will be computed, and will serve as the campaign's unique ID). Make sure to keep that name different from other campaigns to prevent mixing them up.
    - Its corpus. This is where you pick the *Audio Corpus* (previously put in Seshat's `corpora/` folder).

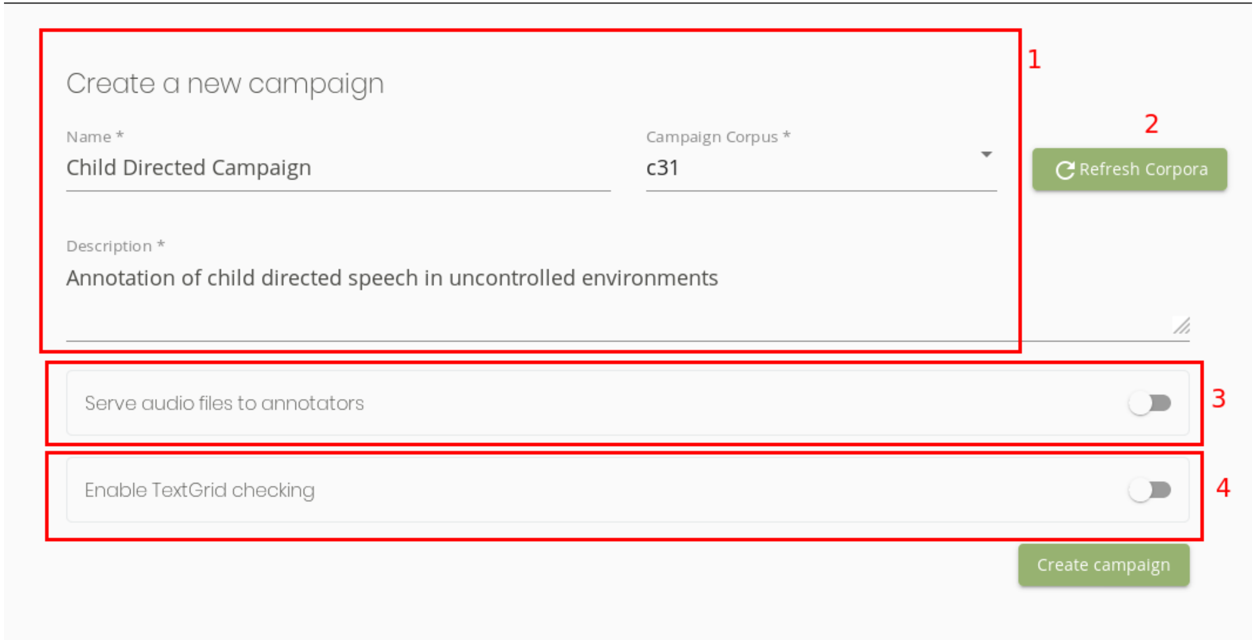


Fig. 1: The campaign creation interface.

- Its description. This has no particular effect of the behavior of Seshat. It's just there to remind you what this campaign is about. Keep it short and informative.

2. This button asks Seshat to refresh the list of detected corpora. Seshat will go over all the files in its `corpora/` folder and if new corpora are detected, will start to track them. This might be useful if you recently added a corpus to Seshat, and it hasn't been automatically detected. **If an added corpus is large, lots of files or big audio files, this make take a bit of time**.
3. This is to set whether or not the annotators will be given the corpus's audio file when being assigned an annotation task. Note that they'll have only access to the audio files they've been assigned to.
4. This is to set the *Textgrid Checking Scheme* for that campaign. Proceed to that section to understand how it works.

6.2.2 Campaign Properties

Once a campaign is created, you'll be able to see it in the campaigns list. You can view a full summary of a campaign's properties by clicking on the *View* button. This is what you should see:

Let's go over the different Informations that are displayed in this view:

- **Tasks** : Number of completed *Annotation Tasks* vs total number of assigned tasks. This progress bar is here to help you track the advancement of your campaign.
- **Files** : Number of files with an assigned *Annotation Tasks* vs total number of files in the campaign's designated corpus. This is to give you a sense of how much of your corpus has been covered by the campaign.
- **Description** : This is the corpus description set at the campaign's creation.
- **Corpus** : This is the name of the corpus that you picked at the campaign's creation. All tasks assignments will be on files from that corpus.
- **Annotators** : This is a list of all the annotators that have received *Annotation Tasks* in this campaign. This is automatically computed by Seshat.

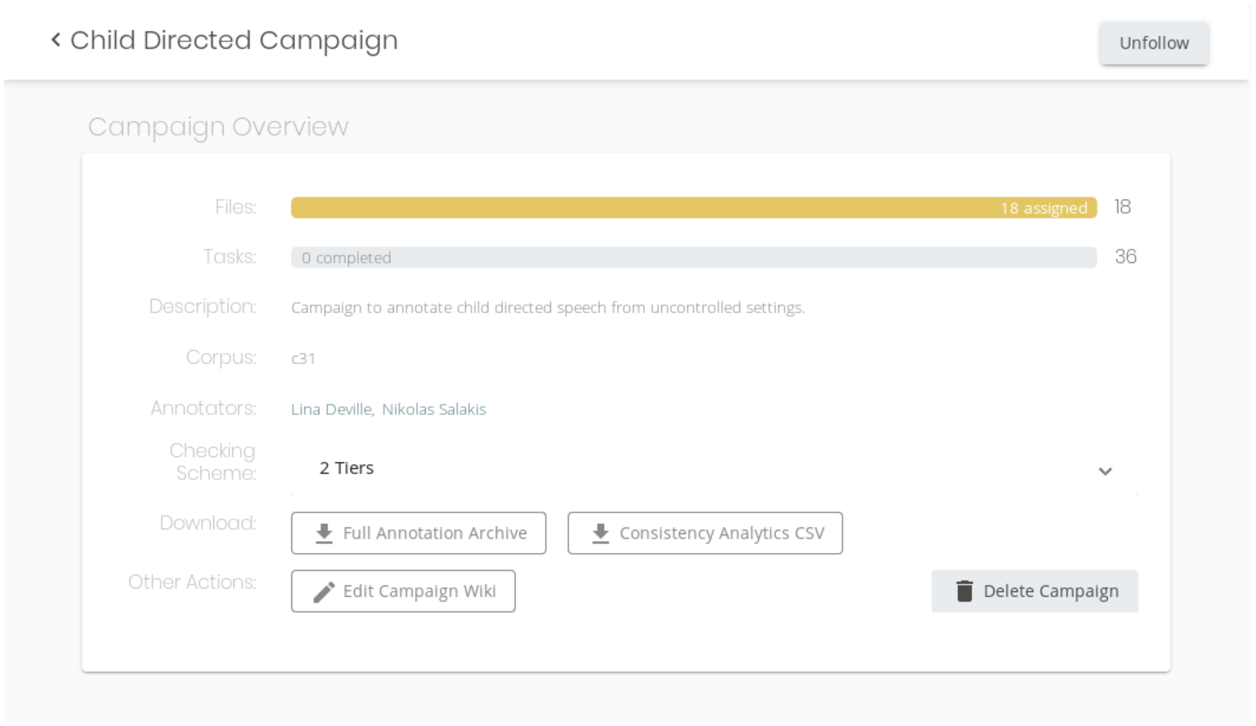


Fig. 2: The campaign status

- **Checking Scheme** : A short summary of this campaign's *Textgrid Checking Scheme*, if you set any.
- **Actions** : Button to retrieve a structured archive (zip file) of all the task's textgrids, along with a CSV summary of the TextGrid's creation.
- **Other Actions** : For now, the only possible action is to edit the campaign's wiki page, which we will explain in the next section.

Note: Apart from the wiki page's content and assigning tasks, none of the campaigns' properties can be edited. The most important field where this is enforced is the campaign's *Textgrid Checking Scheme*. This is to ensure that all TextGrid files submitted to a campaign are "cast in the same mould". If you wish to use another (even slightly different) checking scheme, create another campaign with a new scheme.

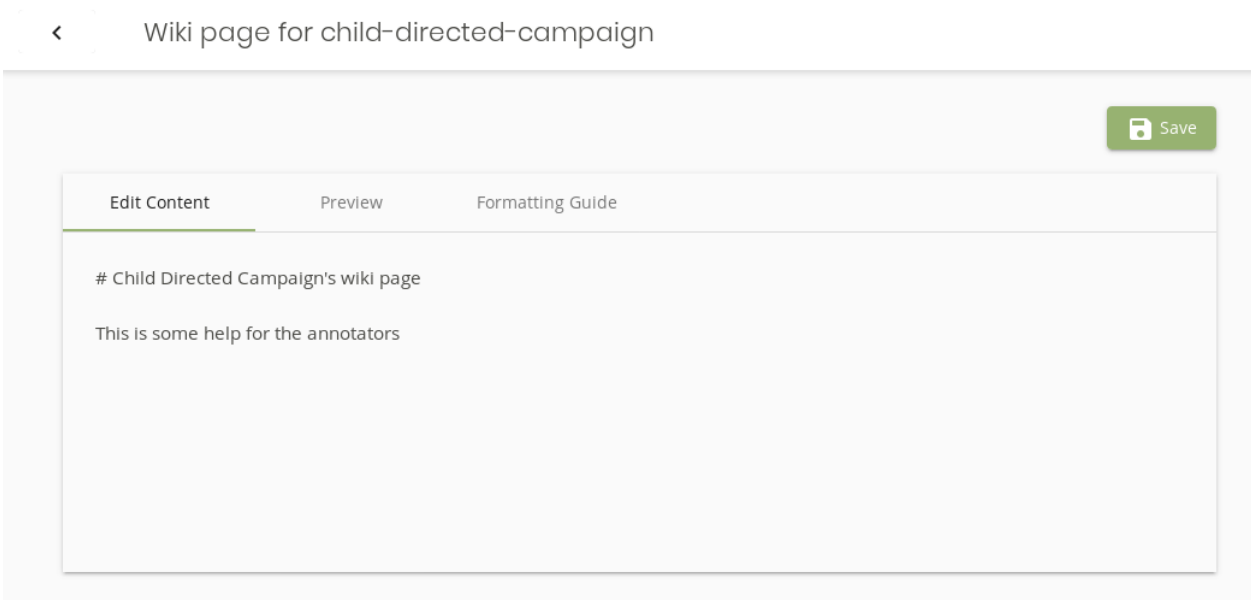
6.2.3 Campaign Wiki Page

Each campaign has a wiki page which can be edited in Seshat's interface (by clicking on the aforementioned "Edit Campaign Wiki" button, in the campaign's status page). Wiki pages are edited using a markup langage called *Mark-Down*.

The campaign wiki page is aimed at being a useful help page for annotators working on *Annotation Tasks*. They can access it when working on an annotation task, from their own interface, along with an automatically generated description of the *Textgrid Checking Scheme* their files should conform to (if there's any such scheme for that campaign) . You should use this as a way to provide them with instructions on how to annotate the campaign's audio files.

Let's go over this interface's functionalities:

- **Wiki Content** : this is the wiki's MarkDown code. This is where you should edit the wiki page's content.



Child Directed Campaign's wiki page

This is some help for the annotators

Fig. 3: The campaign's wiki page edition form.

- **Preview** : This is where you see how the wiki page is rendered to the annotators.
- **Formatting Guide** : this is a short cheat sheet on how to use the MarkDown markup language. We kept it pretty short, thus, if you want to do some more complicated things with it, you should refer to the official documentation

6.3 Textgrid Checking Scheme

A set of rules defining the TextGrid files' structure and content of the annotations. It is set at the beginning of the *Annotation Campaigns*'s creation, and is used to enforce that all TextGrids from the campaign contain the same amount of Tiers, with the same names. It can also enforce, for certain chosen tiers, a set of valid annotations.

Let's take a look at the campaign creation's interface, where a TextGrid checking scheme is specified to explain what each option does:

Let's go through the different parts of this interface:

- Here, we specified **two** tiers. You can specify as many as you want, but it's advised to keep it under 6, as *Double Annotators Tasks* tend to get hard to annotate in Praat's interface when there are more. You can duplicate tiers using the two buttons at the bottom of each tier specification. It is important to note that:
  - If an annotator submits a TextGrid with some tier that is not specified in the checking scheme, it'll be rejected.
  - If an annotator submits a TextGrid with duplicate tiers (two tiers with the same name), it'll be rejected as well.
- Going through the different fields in a tier's specification:
  1. **Tier name**: this is the expected name of the tier in the TextGrid file.
  2. **Validation Option**: Can either be "None", "Categorical" or "Parsed".
     - "**None**" indicates that non-empty Intervals of this Tier won't be validated in any way. Basically, the *Annotators* are free to annotate in any way that is instructed to them for that Tier.

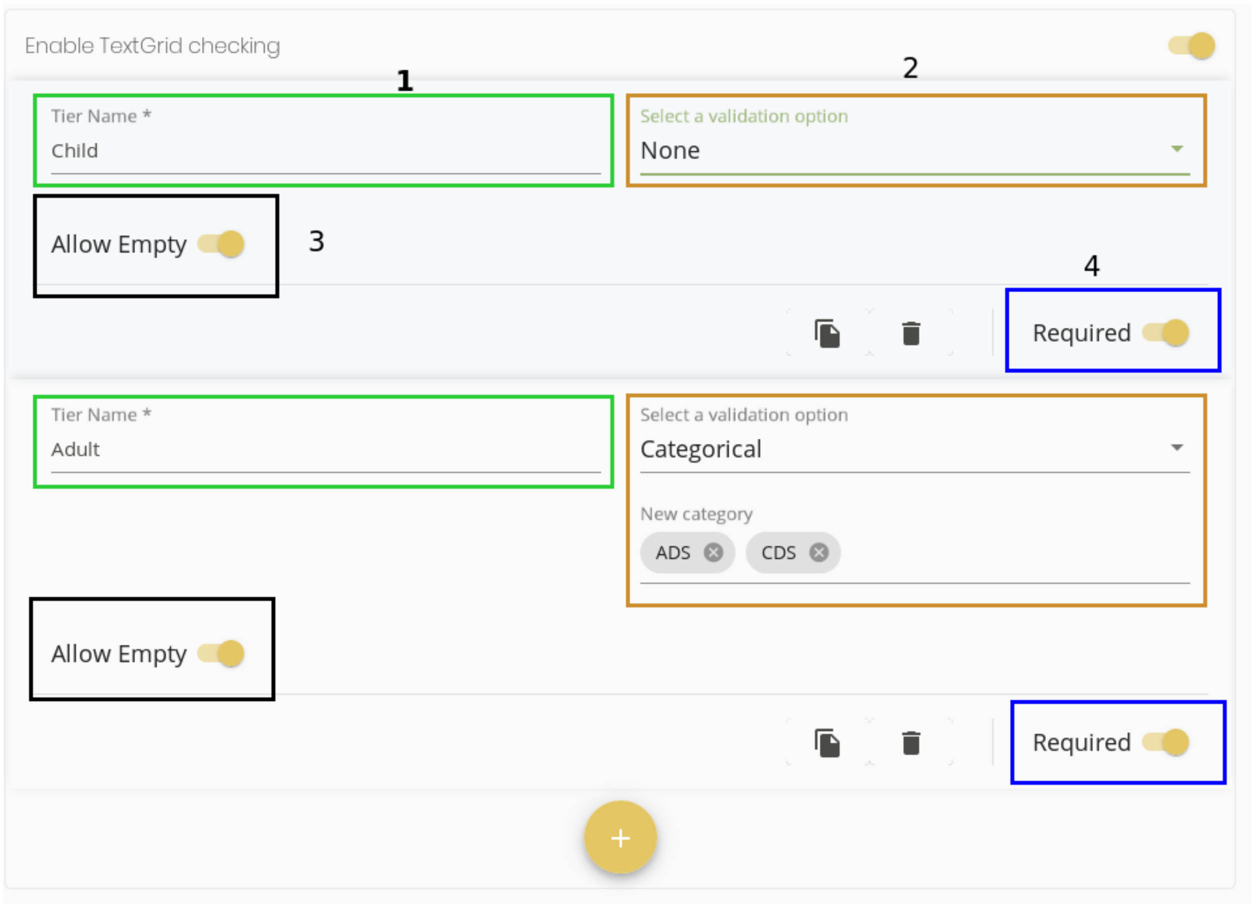


Fig. 4: The Texgrid Checking Scheme specification interface

   - “**Categorical**” indicates that the non-empty Intervals from that Tier can only be part of a set of categories. Here, in the “Child” Tier, we’ve set two categories: *ADS* and *CDS*
   - “**Parsed**” indicates that the non-empty intervals from that Tier will be checked by a custom parser. To learn how to include and use such a Parser, refer to *Custom Parsers*.
3. **Allow Empty**: this sets if Intervals can be empty in that tier. Any Interval between two frontiers that doesn’t contain an annotation will be rejected if this is isn’t set to true. This is particularly useful when you want to cut your audio files in different parts, corresponding to, for instance, different tasks.
4. **Required**: this sets if the Tier is required for the Textgrid to be valid. Deactivating this option will authorize annotators to remove that tier if they feel that it’s not needed on some audio files.

Once the campaign is created (along with its checking scheme), all tasks will be subject to the specified checking scheme. You can understand it as a standard that is enforced on the campaign’s TextGrid files. In other word, any annotated TextGrid file from that campaign will have to be consistent with that standard.

Note: Upon task assigment, a “template” TextGrid file will be generated by Seshat, automatically containing all the specified tiers from the checking scheme. Indeed, they won’t have to create tiers themselves (only delete them if the tiers are not required and they think it’s not need for that task’s audio file).

6.4 Annotation Tasks

It is contained in an *Annotation Campaigns*, it references an audio file from the campaign’s designated *Audio Corpus*}, and assigned to *Annotators*. It can either be a *Single annotator Task* (assigned to one Annotator) or a *Double Annotator Task* (assigned to two annotators, who will annotate the assigned task in parallel).

You maybe noticed that tasks (like annotators’ accounts) can be *locked*. This effectively “freezes” the task, preventing its annotators from submiting new TextGrid files or new comments.

6.4.1 Single Annotator Tasks

This is the ‘default’ task annotation schema. It has three steps:

1. The annotator is given a template file (potentially in an archive also containing the audio file, if specified at the campaign’s creation). If an *Textgrid Checking Scheme* is defined for that task’s parent annotation campaign, the template is pre-filled with the specified tiers.
2. The annotator then has to annotate the file. They can validate it at any moment, and any uploaded TextGrid is logged by Seshat (for any potential later review). Any submission containing errors (and thus not in compliance with the *Textgrid Checking Scheme*) is rejected.
3. Upon the first valid submission, the valid TextGrid file is saved as the “final” TextGrid, and the task is automatically marked as “done” by Seshat. However, if the annotator (or the annotation manager) judges that their work is incomplete, they can still submit corrected version of that final TextGrid file, that will overwrite the last correct “final” TextGrid.

6.4.2 Double Annotators Tasks

This scheme is used to measure inter-annotator (or “rater”) agreement and have a more robust annotation of a single file. Note that it is however much longer to annotate in this way, and should usually be used for a small part of your corpora, as a way to measure how much your annotators agree on these annotations.

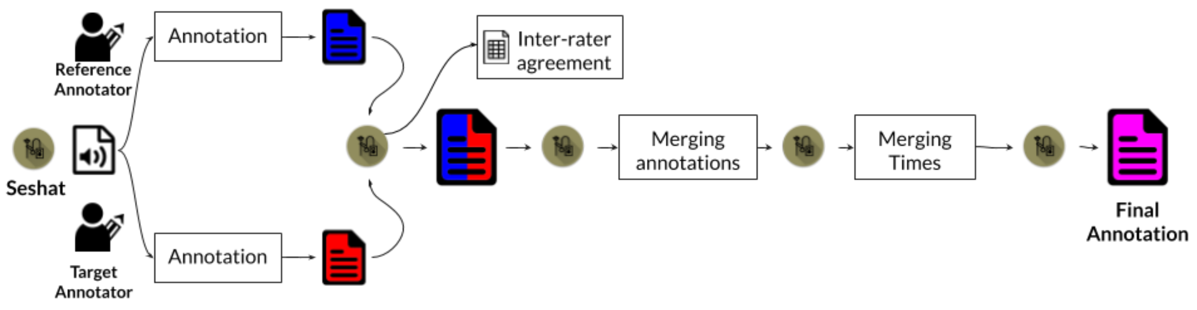


Fig. 5: Double annotators task overview.

6.5 Campaign Manager

Users with the rights to create *Annotation Campaigns* and *Annotators* user accounts, and assign *Annotation Tasks* to *Annotators*.

Campaign managers have access to all campaigns created on Seshat, even those created by other managers (i.e., we tought it best not to have access restrictions for seshat's typical usecase, among small teams of people).

Campaign managers can also **subscribe** to a campaign by clicking on the "Follow" button on a campaign's page. They will, in consequence, receive the notifications from then following events (in that campaign):

- When an annotator completes a task
- When an annotator submits a comment on a task

6.6 Annotators

Users who are assigned a set of *Annotation Tasks*. Their job is to complete the annotation of the audio files with the Praat software. If the TextGrid file they submit does not comply with their *Annotation Tasks*'s *Textgrid Checking Scheme*, Seshat pinpoints their annotation errors with detailed error messages. The annotator can re-submit the affected file to the platform based on these feedbacks.

As you might have noticed in Seshat's client interface, it's possible to lock annotators account. This is to prevent annotators from logging-in to their account and performing any actions, without actually removing the account. This might be useful for two potential reasons :

- if that user was involved in an annotation campaign, and you want to make sure that they don't change the state of your annotated files in any possible way.
- if that user was in a former campaign that you still want stored in Seshat, but for security reasons, don't want to be able to reach the Seshat interface anymore.