# RCPCH Epilepsy 12 - Documentation

**None**

*The Royal College of Paediatrics and Child Health*

# Table of contents

DOI 10.5281/zenodo.6549072

# 1. Welcome to the RCPCH Epilepsy 12 documentation

## 1.1 Introduction

The RCPCH Audit Engine is a framework for national clinical audits. Its first deployment is as a new platform for the RCPCH's established **Epilepsy12** audit, a national audit for childhood epilepsies which has been in place since 2009, but it is designed so as to be reusable for other audits in the future.

National clinical audits are there to collect diagnosis and care process data on patient cohorts with a diagnosis in common, nationally, to benchmark the standard of care and feed back to care-giving organisations about their performance. They are a way to make sure that clinics are meeting centrally-set standards, and give clinics feedback on how they are doing. Most national audits such as Epilepsy12 are commissioned by NHS England.

The RCPCH Audit Engine is a Django 4.0 project, using Semantic UI for the user interface framework. It aims to standardise those elements of a national audit that can be standardised, such as the concept of a Case (patient), Registration of cases to the audit, and management of researchers and administration users.

## 1.2 Stated Aims of the Audit

- Continue to measure and improve care and outcomes for children and young people with epilepsies
- Include all children and young people with a new onset of epilepsy,
- Enable continuous patient ascertainment,
- Use a pragmatic and concise dataset,
- Incorporate NICE Quality Standards alongside metrics about mental health, education and transition to adult services,
- Provide services with local real-time patient- and service-level reporting.

## 1.3 Quality Improvement

- Supporting regional and national quality improvement activities
- Epilepsy Quality Improvement Programme (EQIP)
- Involving children and young people

## 1.4 There are 12 key performance indicators

1. Input into care from a paediatrician with expertise in epilepsies,
2. Input into care from an epilepsy specialist nurse (ESNs),
3. (a) Appropriate tertiary input into care, and
   (b) appropriate epilepsy surgery referral
4. Appropriate first paediatric assessment,
5. Recorded seizure formulation,
6. Access to electrocardiogram (ECG),
7. Access to magnetic resonance imaging (MRI),
8. Accuracy of diagnosis,
9. (a) Discussion of the risks where sodium valproate is used in treatment for girls aged 9 and over, and
   (b) girls and young women prescribed sodium valproate
10. Comprehensive care plan that is updated and agreed with the patient,
11. Documented evidence of all key elements of care planning content,
12. Record of a school individual healthcare plan.

Feedback and Feature Requests

# 2. Clinician User Guide

## 2.1 Overview

### 2.1.1 Clinician Users

This section is to be developed alongside application functionality

# 3. Admin User Guide

## 3.1 Overview

### 3.1.1 Admin Users

This section is to be developed alongside application functionality

# 4. Development

## 4.1 Getting started

### 4.1.1 Python and Django

The RCPCH chose to use Python for developing the Digital Growth Charts and the Epilepsy12 platform, this was because it is an accessible yet trusted language, with an established reputation and userbase.

Django is a web framework for Python, which helps with developing a database-backed web application such as the E12 platform.

In order to develop the platform you should have some familiarity with both of these technologies. Numerous online and free learning resources are available. FreeCodeCamp has a full Python video course, and CodeCademy offers an interactive onlince course. Many other similar courses are available, often for free online. Django itself has a great tutorial here.

### 4.1.2 Git and GitHub

We use Git for local and remote version control, and GitHub to host our source code. We make use of the GitHub Issues feature for tracking our roadmap, features, and bugs. We use GitHub Pages for publishing some of our websites. We also use GitHub Projects to plan work.

We make use of Git branches extensively to manage parallel and concurrent workstreams. Development should always be done in a new branch and only merged with the live branch once suitable testing, review, and authorisation has occurred.

Deployment of our source code is automated where possible, mostly using GitHub Actions, in which a workflow file determines the deployment script, ensuring repeatable and dependable deployments.

### 4.1.3 Open Source

The entire output of the RCPCH Incubator development team is open source. We firmly believe in the importance of open source especially where it applies in the medical domain, which is a special humanitarian case in software terms. Read more about this philosophy here - Open Source is The Only Way For Medicine

### 4.1.4 Code Quality

Open source alone is not a guarantee of good safe code and so we adhere to the tech industry's best practices in terms of Python style guidance, linting, and testing.

### 4.1.5 Security

Security is a very important aspect of managing projects such as these, and we use secure passwords, two-factor authentication, signed Git commits, branch protection, and keep all credentials in environment files. See The Twelve-Factor App for the absolute textbook on this.

## 4.1.6 Legal

Details of Information Governance, Clinical Safety and Medical Device registration can be viewed in the Legal section.

# 4.2 Docker dev setup

In order to simplify the development environment setup and provide greater consistency between development and production environments, we have packaged the application as a Docker image. This means that you don't need to worry about conflicts of Python versions, Python library versions, or Python virtual environments. Everything is inside the Docker container.

## 4.2.1 Setup for development using Docker Compose

Install Docker on your development machine. Instructions for all supported platforms are here https://docs.docker.com/get-docker/

Clone this repository to your code folder:
`git clone https://github.com/rcpch/rcpch-audit-engine.git`

Navigate into the folder
`cd rcpch-audit-engine`

Start the development environment
`docker compose up`

> **Tip**
>
> Note that the command changed from `docker-compose` to `docker <space> compose` with more recent Docker versions.

This should create a `db` container for the database and another `web` container for the Django app. The `web` container is built with the correct Python version, all development dependencies are automatically installed, the database connection is created, migrations applied and seed data added to the database. The entire process takes less than 30 seconds.

View the application in a browser at http://0.0.0.0:8000/

Changes you make in your development folder are **automatically synced to inside the Docker container**, and will show up in the application right away. This Docker setup is quite new so please do open an issue if there is anything that doesn't seem to work properly. Suggestions and feature requests welcome.

## 4.2.2 Executing commands in the context of the `web` container

You can run commands in the context of any of the containers.

The below command will execute `<command>` inside the `web` container.

```
docker compose exec web <command>
```

For example, to create a superuser

```
sudo docker compose exec web python manage.py createsuperuser
```

## 4.2.3 Running the test suite

```
sudo docker compose exec web coverage run manage.py test
```

## 4.2.4 Shutting down the Docker Compose environment

`^ Ctrl` + `C` will shut down the `web` and `db` containers but will leave them built. You can restart them rapidly with `docker compose up`.

To shut down and destroy the containers so that you can start again from scratch (for example if you want to rebuild and re-seed the database) then use

```
docker compose down
```

## 4.2.5 Tricks and tips.

- Although the `docker compose` setup is very convenient, and it installs all the runtime development dependencies *inside* the `web` container, one thing it can't do is install any *local* Python packages which are required for text editing, linting, and similar utilities *outside* the container. Examples are `pylint`, pylint_django`, etc. You will still need to install these locally, ideally in a virtual environment.

# 4.3 Documentation

## 4.3.1 Introduction

The RCPCH Audit Engine / Epilepsy 12 documentation is made with Material for MkDocs, which is a framework, separate from Django, which takes Markdown source files from `docs` and compiles them into a static HTML site. These static files are then served from our hosting resources.

## 4.3.2 Setup Python and Pyenv

See LINK for a guide on how to set up a Pyenv virtual environment containing the correct version of Python.

The documentation repository contains a `.python-version` file which will automatically select the correct Pyenv for you when you navigate to the repository, as long as you have called your virtualenv `mkdocs`.

You need to have Material for MkDocs installed (this installs MkDocs as well for you)

In most circumstances this is simply `pip install mkdocs-material`, but if you hit issues then there is more information at https://squidfunk.github.io/mkdocs-material/getting-started/

## 4.3.3 `mkdocs serve`

`mkdocs serve` starts up a development server which will auto-reload after changes to the source files, and will serve the documentation on `localhost:8001`.

## 4.3.4 How to edit content

- Check out a **new local branch** on which to make the changes, with a **descriptive name**.
- Make changes to the Markdown files in the `docs` folder in the project root.
- Ensure any new or renamed files are listed in the `nav` element within `mkdocs.yml` or they won't show up.
- Save and the changes.
- The auto-reloaded site will show the changes.
- Commit the changes. Try to keep commits tidy and 'atomic' - in that ideally a single commit should be one new or edited piece of content, not a whole raft of changes. This allows us to easily select which commits to include.
- Push the changes and create a pull request to have them included in the main project.

## 4.3.5 Reference guides

MkDocs and Material for MkDocs (the MkDocs theme we are using) have a host of features for making beautiful, practical, functional and easily navigable documentation.

#### Markdown

Fundamental to the way the documentation works is the use of a simple set of text annotations called 'Markdown', which are easily readable and editable as text files but can be compiled into HTML for web viewing. MarkDown is hugely popular across the web for rapid entry of web-native formatted text, being the basis of much of GitHub, StackOverflow, and Discourse's functionality.

Markdown uses characters like asterisks ( `*` ), hashes ( `#` ) and others, to effect its formatting. For example: `**bold**` to denote **bold** text. It's simple to get used to and, once you're used to it, very productive too. One advantage is that formatted text stays where it's been put, unlike with some word processors in which the GUI formatting tools can have you chansing formatting changes all over a document.

#### ONLINE EDITING OF MARKDOWN

If you are new to Markdown editing, you can use GitHub's interface itself to edit online, by clicking the 'pencil' edit icon in the top right corner of any source code page. There are also external tools like Prose.io and StackEdit which give you a nice interface for editing MarkDown online, and will sync the changes with GitHub for you. If you need help getting set up, contact us in the Signal chat.

If Markdown seems daunting then another option is simply to edit the content in the word processor of your choice and then ask one of the RCPCH Incubator team to convert it to Markdown and add it to the documentation.

### Material for MkDocs

On top of the basic features of Markdown, MkDocs and the Material theme adds all the nice website appearance and many additional features for making beautiful documentation sites.

A good overview can be had from looking at the Material for MkDocs Reference section and from copying existing code in the documentation that does what you need.

### MkDocs

If you can't find functionality documented in the Material for MkDocs theme website, this is usually because it is functionality which comes from MkDocs, the underlying framework, itself. See the MkDocs site for these features.

### Pymdownx extensions

Some of the features such as Keys come from extensions like Pymdownx

# 4.4 Manual dev setup

If you prefer to set up a development environment manually, here are the steps. Please note that we do not provide support for developers using a bespoke setup, only the Docker development environment is supported.

This manual approach will require you to have much more familiarity with configuring PostgresQL, Django, and Python to achieve your aims, and is not for beginners.

## 4.4.1 Install PostgreSQL and create the database with the correct credentials

You will need the Postgresql database, which can be installed natively on your development machine, or (recommended) can be installed using Docker.

Using the command below will create a development database with credentials that match those in our `example.env` file.
You will need Docker to be installed on your local machine. (Please search the web for instructions for installation on your operating system and setup)

```
docker run --name epilepsy12postgres \
    -e POSTGRES_USER=epilepsy12user \
    -e POSTGRES_PASSWORD=epilepsy12 \
    -e POSTGRES_DB=epilepsy12db \
    -p 5432:5432 \
    -d postgres
```

## 4.4.2 Install the correct Python version

If you don't have python 3.10 installed already, you will need it.

We recommend the use of a tool such as pyenv to assist with managing multiple Python versions and their accompanying virtualenvs.

```
pyenv install 3.10.0
```

On some platforms, you will get errors at build-time, which indicates you need to install some dependencies which are required for building the Python binaries locally. Rather than listing these here, where they may become out of date, please refer to the pyenv wiki which covers this in detail.

Then create a virtual environment:

```
pyenv virtualenv 3.10.0 rcpch-audit-engine
```

Clone the repository and `cd` into the directory:

```
git clone https://github.com/rcpch/rcpch-audit-engine.git
cd rcpch-audit-engine
```

Then install all the requirements. Note you can't do this without PostgreSQL already installed first.

```
pip install -r requirements/development-requirements.txt
```

## 4.4.3 Initialize the environment variables

```
source example.env
```

**Danger**

The included example environment variables are not secure and must never be used in production.

## 4.4.4 Prepare the database for use

```
s/migrate
```

## 4.4.5 Create superuser to enable logging into admin section

```
python manage.py createsuperuser
```

Then follow the command line prompts to create the first user

Further users can subsequently be created in the Admin UI

## 4.4.6 Running the server

Navigate to the epilepsy12 outer folder and run the server:

```
s/runserver
```

or

you may need to allow permissions to run the bash script in that folder first:

```
chmod +x ./s/runserver
chmod +x ./s/migrate
chmod +x ./s/seed
chmod +x ./s/init
```

## 4.4.7 Seeding the Database

You will need to see the hospitals table with hospitals from the Constants folder.

```
python manage.py seed --mode=seed_hospitals
```

If you need to delete all the hospitals:

```
python manage.py seed --mode=delete_hospitals
```

To add the semiology keywords to the database:

```
python manage.py seed --mode=seed_semiology_keywords
```

To add the some dummy cases to the database:

```
python manage.py seed --mode=seed_dummy_cases
```

If you want to seed with all these, there is a short cut in the start folder:

```
s/seed
```

## 4.4.8 Running the tests locally

We have used the coverage package to test our models. It is already in our development requirements, but if you don't have it installed, install it with `pip install coverage`

Run all the tests

```
coverage run manage.py test
coverage html
```

If the `htmlcov/index.html` is opened in the browser, gaps in outstanding testing of the models can be found.

# 4.5 Code Style

## 4.5.1 IDE / Text Editor

We recommend the use of the popular VSCode editor, which has rapidly become the most popular editor in the last few years. It has good support for Python and Django, and additional features such as LiveShare and Azure integration are also helpful.

The below instructions regarding linting and formatting assume the use of VSCode

## 4.5.2 Linter

We use the PyLint linter. It promotes consistency if all of the team use the same linting and formatting rules.

You may need to install the `pylint_django` plugin and add `--load-plugins=pylint_django` to the PyLintArgs:

- Press `^ Ctrl` + `,` to enter VSCode's Settings
- Search for `python linting` and scroll down to PyLint
- Ensure `Python > Linting: Pylint Enabled` is checked
- In `Python > Linting: Pylint Path` write `pylint_django`
- In `Python > Linting Pylint Args` add `--load-plugins=pylint_django`

## 4.5.3 Formatter

## 4.5.4 Imports

Python imports should be categorised:

```
# standard imports

# third party imports

# RCPCH imports
```

In addition, both packages and individual functions/classes should be listed alphabetically.

All of the above measures help to prevent duplicates, ensures tidiness and maintainability, and lets us see easily which of our imports are most reliable and trusted.

# 4.6 Project Design

The RCPCH Incubator development team used Django 4.0 as it is a python (3.10) framework which is mature, accessible and well-documented. It is founded on the concept of a project which can have multiple applications within it. This meant that RCPCH could have an audit-engine project within which multiple audit applications might sit, sharing resources, for example relating to authorization and authentication, or potentially constant values and so on. RCPCH administers several national audits on behalf of children and their families and the paediatric organisations that serve them, so Django offered the opportunity in future to bring together audits into a single platform.

The top level folder, therefore, is `rcpch-audit-engine`, which contains the `settings.py`, project `urls.py` as well as `asgi.py` and `wsgi.py` files.

Within the rcpch-audit-engine platform, currently epilepsy12 is the only application.

## 4.6.1 Application Structure

An application folder sits within the `rcpch-audit-engine` folder and is named after the application name (`AppName`).

**The key files are:**

- `apps.py`: This subclasses the Django `AppConfig` and it is here that the application name is set
- `admin.py`: It is here that the custom user is set as `Epilepsy12User`, personalizations to the admin interface are set, and rules for user creation including the `createsuperuser` function.
- `decorator.py`: Decorators are python functions which wrap a function. They are used to protect routes by performing validation and authorization, redirecting as necessary to 404 and 403 screens.
- `forms.py`: Forms in Django automate much of the complication of webforms. In the Epilepsy12 project, the power of Django forms has not been leveraged for reasons discussed further down. There are 2 types of Django form that have been retained. The first are the login/account creation/registration forms. Custom forms have been created to match the Epilepsy12 and RCPCH styles, but the form validation logic sits on top of Django. The other place Django forms have been used is in Case creation/updating within the Epilepsy12 application. There is a file for each form in the `forms_folder` at the top level within the Epilepsy12 application. These in turn are imported into `forms.py`.
- `models.py`: The structure of each Model Class can be found in the `models` folder, at the top level within the Epilepsy12 application. There is a file for each model class. These in turn are imported into `models.py`.
- `serializers.py`: This file is part of the `django-rest-framework` and is analagous to forms in the application. Serializers take in the models to generate the API definitions.
- `urls.py`: All routes, both for the application and the API, are defined here and imported into project-level `urls.py` of the same name in the `rcpch-audit-engine` folder above. The routes use the Django `path()` function which link defined paths and wildcards with functions in `views.py`. Routes for the API use the `django-rest-framework` `Router` module.
- `Views.py`: All views can be found in `view_folder`, at the top level within the Epilepsy12 application. There is one for each model, with the exception of `multiaxial_diagnosis_views.py` which has functions for all its related models (`Episode`, `Syndrome` and `Comorbidity`). `views.py` imports all views from this folder.

**The folders are:**

- `constants`: This contants multiple files each containing constant values used in the models and elsewhere. The follow the standard that uppercase is used to denote its immutable nature.
- `coverage_tests`: Tests are found here and collected automatically by the Coverage package and executed on the command line with the command: `coverage run`.
- `forms_folder`: see `forms.py` above.

- `general_functions` : this folder contains files for functions that can be called across the application. It contains functions particularly for external API calls (for example to the SNOMED server), but also houses the logic behind fuzzy matching for the description words in the DESSCRIBE tool.
- `management` : Files here contain functions that are run from the command line with the prefix `python manage.py` . These include functions that seed the database with dummy data in development.
- `migrations` : This is a Django folder and should not be touched unless the user is a confident Django user. Any changes to the models are stored here and therefore the history in this folder reflects the history of the database design over time. It should always be checked into Git.
- `models_folder` : see `models.py` above.
- `templatetags` : files in this folder contain helper functions used by Django templates.
- `view_folder` : see `views.py` above.

## 4.6.2 Folders outside of the Applications

There are several other top level folders and files

#### Files

- `CITATION.cff` : This has been generated by zenodo and allows the `rcpch-audit-engine` project to be cited in the academic literature
- `docker-compose.yml` : see manual setup
- `dockerfile` : see manual setup
- `example.env` : Environment variables for security - these are an example only and in production should be hidden.
- `LICENSE` : This is the AGPL-3.0 License.
- `manage.py` : This is a django file and should not be touched.
- `README.md` : This is a readme file but contains very little as redirects to this documentation site.
- `requirements.txt` : This is a python file and contains a list of all the dependencies +/- versions. The RCPCH Incubator team are judicious with dependencies and these should only be used if the dependency is established, well-documented and meets an important need where building in-house would be impractical. The file references the `requirements` folder which contains dependencies organised into folders based on whether they are installed for development or production. The syntax for installation is: `pip install -r requirements/common-requirements.txt`

#### Folders

- `s` : This is an RCPCH incubator standard. Shortcuts related to running the project or migrating or seeding the database are here. The commmands follow the example: `s/migrate` . Any custom functions are found in the `manage` folder.
- `static` : contains all static files. The RCPCH Incubator recommend any css files be persisted here, rather than using CDNs to protect from situations where data signals are not available. This has a small latency cost, and involves regular review of dependencies to ensure they are up to date.
- `tables` : This has no function and is listed for deprecation. It contains olds excel sheets of database models in the previous version of Epilepsy12
- `templates` : All templates are found here.

# 4.7 Epilepsy12 Database

## 4.7.1 Frameworks

The platform has been written in Django 4.0 with a Postgresql Database backend.

## 4.7.2 Structure

Database structure largely follows the elements of the order. All the models largely have a one to one relationship, with some exceptions. The models are as follows:

- **Case**: There is one record for each child in the audit
- **Registration**: The is one registration for each case
- **FirstPaediatricAssessment**: This a key milestone in the audit - the date of the first paediatric assessment triggers the initiation of the audit for that child. There can only be one assessment per registration.
- EpilepsyContext: The fields in this model describe the risk factors for epilepsy for each child in the audit. There can be only one record in this model per registration.
- **MultiaxialDiagnosis**: This is the formulation of the child or young person's epilepsy and describes their epilepsy in a multiaxial way using the DESSCRIBE approach. There can be only one multiaxial diagnosis record per registration.
- **Assessment**: This is termed Milestones in the audit and in due course the model and its related views will be refactored. It relates particularly to dates and location of key caregivers: in particular the consultant paediatrician with expertise in epilepsy, the paediatric neurologist, the children's epilepsy surgery centre if eligible and the paediatric epilepsy nurse specialist. There is one record in the Assessment model per registration.
- **Investigations**: This stores information on whether key investigations have been performed in a timely way and covers ECG, EEG and neuroimaging such as CT or MRI. One record in the Investigations model exists per registration.
- Management: This stores information about medications and individualized care plans for each child in the audit. There is only one record in the Management model per registration.
- **Episode**: The Episode model records information about each seizure-type. A child's epilepsy may compromise multiple different seizure types, some of which are epileptic, some of which are not. One record in the MultiaxialDiagnosis model can therefore have multiple Episode records. For the MultiaxialDiagnosis record to be complete, there must be at least one associated Episode record which is epileptic.
- **Syndrome**: A child's epilepsy can be part of a broader syndrome or syndromes. The Syndrome model stores information about date of diagnosis and syndrome name. It is possible for a child to have more than one Syndrome associated with their epilepsy, therefore there is a one to many relationship between MultiaxialDiagnosis and Syndrome models.
- **Comorbidity**: The Comorbidity model captures information principly on development, educational and behavioural comorbid diagnoses a child may have. Since it is possible to have more than one, one record in MultiaxialDiagnosis can have several records (or none) in the Comorbidity model.
- **AntiEpilepsyMedicine**: This model has a many to one relationship with the Management model. One child may be on more than one medicine, and the medicines maybe used either for the epilepsy or as rescue for a seizure.
- **Site**: The relationships here are complicated since one child may have their epilepsy care for different things in different Hospital Trusts. Each Case therefore can have a many to many relationship with the HospitalTrust trust model (since one HospitalTrust can have multiple Cases and one Case can have multiple HospitalTrusts). The Site model therefore is a link model between the two. It is used in this way, rather than relying on the Django built-in many-to-many solution, because additional information relating to the hospital can be stored per Case, for example whether the site is actively involved in epilepsy care and what service it provides (acute paediatric, tertiary neurology or epilepsy surgery care).
- **HospitalTrust**: This model stores information about each hospital in England, Scotland and Wales. It is used as a lookup for clinicians as well as children in Epilepsy12. It has a many to many relationship with Case and a many to one relationship with Epilepsy12User. It is seeded from the `constants` folder with a `` `JSON `` list of hospital trusts.
- **Epilepsy12User**: The User base model in Django is too basic for the requirements of Epilepsy12 and therefore a custom class has been created to describe the different users who either administer or deliver the audit, either on behalf of RCPCH, or the hospital trusts.
- **Keyword**: This model stores the keywords that are used to describe the semiology of each seizure event. The original list is taken from the International League against Epilepsy 2017, but is actually badly in need of enrichening. Even the word 'shaking' is missing. Part of the Epilepsy12 project is to validate the description of a seizure using keywords stored in this model. The original list of words is seeded on first run from the `constants` folder.
- **Group**: Not strictly an Epilepsy12 model, but a Django model tied to the User class. There are 6 custom groups (3 RCPCH, 3 hospital trust) with differing levels of access depending on status. The permissions, which are granular and relate to the individual model fields, can then be allocated to groups, allowing admin staff to ensure that permissions are granted in a systematic way.

## 4.7.3 Migrations

Any changes to the database structure are captured in the migrations, and this is run at each deploy, with any fresh migrations being applied if present at that point. They are stored in the `migrations` folder and these files should not be altered and should be checked into version control.

# 4.8 Design Decisions and Rationale

A specific design decision was made not to leverage the significant time-saving power of Django's class-based views and forms, in favour of HTMX, so some explanation is needed here to explain the rationale.

A criticism of the previous imagining of the Epilepsy12 audit, and the rationale to rebuild, was that it was too detailed and complicated. This meant users (largely clinicians) filling in the audit fields had significant audit fatigue completing the many screens of questions (which were verbose and detailed), with the consequence that many did not engage fully with the audit, or much of the information was incomplete. In planning therefore, the RCPCH Incubator development board and Epilepsy12 Project board teams agreed to reduce the burden on clinicians in 3 ways:

1. The Project Board to review all questions and remove any that were not essential to audit aims or key performance indicators. All fields would therefore be mandatory.
2. The Development Board to build 3 interfaces to Epilepsy12:
3. An API interface either for automated population of some or all fields, particularly demographic information such as child's NHS number, name and postcode.
4. A renewed frontend interface for clinicians to enter information easily if they were unable to use the API.
5. A separate frontend interface for children and families to review all data held on them and their care, that they might be offered the opportunity to approve its accuracy and provide final consent to its inclusion in the audit. Whilst legally this is not needed, ethically it was felt important to offer this to families who were entrusting their data to be used to improve services for all.
6. The Design team together with the Development Board to ensure that any frontend application would be designed to be intuitive and navigable, that each entry be completed quickly and easily.

## 4.8.1 Front End Design

RCPCH have a number of microsites on paediatric topics. To harmonise with the branding of these, Epilepsy12 was to adopt the branding of RCPCH to take advantage of the familiarity that existing users (many of whom are paediatricians) would already have with RCPCH resources.

#### Reactivity

An important consideration for the Development Board was that any frontend application should therefore be reactive: Django applications typically require the user to complete multiple fields in a form before a submit button is pressed which POSTs the form contents to a form model instance for validation and then persistence in the data model. This is not reactive and this was felt to be counter to the stated aims of the project board that the forms be quick and easy to fill. It also makes it more difficult for different fields to be completed at different times throughout the audit year as different milestones were completed, allowing the user to fill the audit out in any order as things happened, without having to rely on all pieces of required information being present before form submission. This underlined the idea that audit is not linear, and no two children's epilepsy journeys are the same.

##### HTMX

HTMX is a small javascript library that uses allows the developer to access the DOM without writing javascript. The core of the library is that it allows the user to issue AJAX requests directly from HTML, facilitating server side rendering of partial html templates rather than triggering a page reload on submit.

This meant that Epilepsy12 views could receive individual POST requests for each field in the form and return small template partials allowing a save-as-you-go approach. Forms would be easy to customize and design, would be reactive, smooth and quick to complete for the user. The user should be able to save information at the time it is known and not be dependent on information required by different fields in the same models which might only be known at different times. For example, the MRI request date might be known before the date it has been reported, so the user should be allowed to record that date and record the report date later on. This would allow the user to complete fields not necessarily in order, but instead as things happened, and save all

information immediately without having to submit a form and wait for the page to refresh with the new information. The downside was threefold:

- Class-based views and forms could not be used, since validation would have to be based on individual fields, not the full instance of the model.
- No fields in the model could be required to save a whole instance of the model.
- There needed to be a route for every question in the audit, with a corresponding function in `views.py`

#### Templates

Templates are all written in django-html. They are organized as follows:

- `epilepsy12` : all templates in this folder relate to the Epilepsy12 application webapp
- `registration` : all templates in this folder relate to signing in and signing up
- `rest_framework` : this subclasses the `api.html` template used in the API webview.

Within the `epilepsy12` folder templates are organized according to the form they represent or some element of the form.

##### BLOCKS AND PARTIALS

These form a large part of the frontend design, since for the page to be reactive, it has to be broken into elements which can refresh without touching the rest of the DOM.

- `base.html` : This forms the base of all content and is used as a parent template for most content. It includes the `<head>` tag and any javascript/jquery on which semantic ui is dependent.
- `audit_section.html` contains the structure of each form and has blocks within it for the text of the headers and footers which contain title, child identifiers if needed and basic information about form completion.
- `partials` : This subfolder contains within it subfolders, one for each form. The form is broken up into partials which contain a single field or small number of fields which can be persisted and updated together.

##### PAGE_ELEMENTS

This is a subfolder of `partials` and needs extra explanation, as this contains the customized widgets that make up the RCPCH audit forms. The widgets are named as follows:

- check_box_group.html
- date_field.html
- frida_select.html
- hospitals_select.html
- multiple_choice_multiple_toggle_button.html
- radio_button_group.html
- select.html
- single_choice_multiple_toggle_button.html
- toggle_button.html
- snomed_select.html

These are all semantic ui elements, but have been customized to RCPCH design standards. Incomplete fields are rendered as blue round target icon next to the incomplete field, with a tooltip messaging this on hover, whilst completed fields are shown as pink ticks.

The parameters for the elements are not all the same but follow the same broad pattern: *hx_post*: the url posted to including parameters *hx_target*: the id of the html element to target the server response *hx_trigger*: usually 'change' but can be any event *label*: this is the label text (accessed from the help text in the model) *reference*: this is the reference text (accessed from the help text in the model) *date_value*: date value *data_position*: this is the position of the popup label (js independent) ['top left', 'top center', 'top right', 'bottom left', 'bottom center', 'bottom right', 'right center', 'left center'] *input_date_field_name*: **date_field only** the name of the input element for the date_field - should match the field name being updated *disabled*: a flag to reflect if the

element is enabled or not *hospital_list*: **hospital_select only** a filtered list of hospitals *test_positive*: common to most of the elements. It is usually the name of the field in the model to be updated *hx_field_list*: **check_box_group only** list of radiobutton options *hx_field_list_name*: **check_box_group only** the name of the list of options *hx_model*: **check_box_group only** model instance to update on selection

#### OTHER ELEMENTS

The following partials are unique:

*steps.html*: This partial is the menu down the left of the screen which signals to the user which form they are interacting with, as well has how far through completion they are. *date_of_first_paediatric_assessment.html*: This partial is found in `templates/epilepsy12/partials/first_paediatric_assessment` and includes a `date_field.html` partial

These are both instances where the htmx custom `trigger` function is used. This is where an element that is updated in one part of the screen leads to the update of another element in another part of the screen. This means the trigger to refresh the second element has to be called from the view which returns an updated instance of the first. The `steps.html` element is updated each time any field in any form is updated. This is done in the view (in `epilepsy12/view_folder/common_view_functions.py`) where the htmx trigger (in this case `'registration_active'`) is attached to the response as follows:

```
# trigger a GET request from the steps template
    trigger_client_event(
        response=response,
        name="registration_active",
        params={})  # reloads the form to show the active steps
```

`'registration_active'` is defined in `views.py`. In this code snippet, the Registration and AuditProgress models are queried and passed on to the `steps.html` partial which is rerendered.

```
# HTMX generic partials
def registration_active(request, case_id, active_template):
    """
    Call back from GET request in steps partial template
    Triggered also on registration in the audit
    """
    registration = Registration.objects.get(case=case_id)
    audit_progress = registration.audit_progress

    # enable the steps if has just registered
    if audit_progress.registration_complete:
        if active_template == 'none':
            active_template = 'register'

    context = {
        'audit_progress': audit_progress,
        'active_template': active_template,
        'case_id': case_id
    }

    return render(request=request, template_name='epilepsy12/steps.html', context=context)
```

The second place that custom htmx triggers are used is in the `confirm_eligible` function of `registration_views.py` where the date of the first paediatric assessment can only be enabled once a primary audit site has been allocated and the user has confirmed the child meets all the eligibility criteria. In the same way, a custom htmx trigger is attached to the response object:

```
# activate registration button if eligibility and lead centre set
        trigger_client_event(
            response=response,
            name="registration_status",
            params={})  # updates the registration status bar with date in the client
```

The `registration_status` trigger calls the function of the same name in `registration_views.py` which returns the `registration_dates.html` partial with an updated instance of registration allowing the date fields to be enabled.

```
@login_required
@group_required('epilepsy12_audit_team_edit_access', 'epilepsy12_audit_team_full_access', 'trust_audit_team_edit_access', 'trust_audit_team_full_access')
def registration_status(request, registration_id):

    registration = Registration.objects.get(pk=registration_id)
    case = registration.case

    context = {
        'case_id': case.pk,
        'registration': registration
    }

    template_name = "epilepsy12/partials/registration/registration_dates.html"
```

```
    response = recalculate_form_generate_response(
        model_instance=registration,
        request=request,
        context=context,
        template=template_name
    )

    return response
```

## 4.8.2 Scores and Progress

User progress for each case is stored in the AuditProgress model which is updated each time a field is created or updated. The logic for this is common to all fields and therefore all logic is held in `epilepsy12/view_folder/common_view_functions.py`.

There are several functions:

#### Response generation

```
def recalculate_form_generate_response(model_instance, request, context, template, error_message=None):
```

This receives a request and model instance from the calling view function with the context and the template and uses these to calculate which fields in the model have scored values (since all fields are initially `None`), and compare this with the number of expected fields for that model instance. This is complicated by the fact that each form is dynamic, and therefore the total number of expected fields depends on user choices. For example, if the child is not eligible for epilepsy surgery, they do not have to complete date of referral and date seen at the local children's epilepsy surgery centre. There are several functions, therefore, that accept a model instance and use this to determine what the user has scored so far, and what the minimum expected number of fields should therefore be for a completed record. This progress is stored in the AuditProgress model which can be used to update the progress wheel in the `steps.html` partial as well as signal to the user if that form has been fully completed. It can also be used to know if all forms are complete and that the child's data can therefore be submitted.

Once progress calculations have been performed and the AuditProgress model has been updated, the response object can be constructed and the `"registration_active"` custom htmx trigger discussed above can be attached before returning to the calling view function.

#### Request validation and model updating

```
def validate_and_update_model(
        request,
        model_id,
        model,
        field_name,
        page_element,
        comparison_date_field_name=None,
        is_earliest_date=None):
```

This function supercedes logic that was originally written as a decorator. Decorators are python functions which wrap another python function. They are typically used to protect routes to prevent unauthorized users gaining access and redirect them to the login or a 403 page.

Originally used in this way to reduce boiler plate code and update the model with the POST request value from the template, it soon became clear that it would not be possible in this way to return error messages to the template. The code was therefore moved here as a simple function to be called by most view functions to prevent code repeating.

It accepts a request, a model and model primary key, as well as the field name to be updated. If a date field is involved and some comparison of dates is needed, that is added an optional parameter. This then runs a validation on the POSTed data and updates the model with the new value. In the event of invalid data, the model is not updated and a `ValueError` is raised with a meaningful message which can be caught in the calling view function through the `try...except...` method and passed back to the template to be shown to the user.

Currently error messages largely exist for dates which are impossible (for example inappropriately in the future or where scans are reported before they have been requested), but in due course this will be extended to all errors beyond those that might be anticipated.

## 4.8.3 API

The Epilepsy12 Team had originally envisaged an API to allow EHR to Audit communication to reduce the burden of field completion on clinicians and administrative teams. For EHRs nationally to integrate with an Epilepsy12 API for audit submission was not realistic for most trusts and the front end user interface remains the main way the audit will be completed. Django Framework has been added though to include some key endpoints to facilitate EHR to Epilepsy12 communication, particularly with respect of case addition. In particular there are 2 endpoints that have been created:

1. `api/v1/register_case` : accepts POST request of an NHS number of an existing case and hospital ID to register an existing case in Epilepsy12
2. `api/v1/add_case_to_hospital_list` : accepts POST request of an NHS Number and HospitalID to create a record of a child in Epilepsy12, associated with a hospital

Other endpoints can be added in due course.

# 5. Legal

## 5.1 Overview

### 5.1.1 Copyright

Copyright is asserted over all RCPCH intellectual property outputs by the Royal College of Paediatrics and Child Health. We will defend this copyright in all territories.

### 5.1.2 Open Source license

The RCPCH Audit Engine is licensed under the GNU Affero General Public License v3.0

All documentation and textual work (such as this documentation site) is licensed under a Creative Commons Attribution-ShareAlike 4.0 International license.

Our licensing arrangements do not affect the licenses of any underlying technologies such as Django (3-clause BSD) MkDocs (BSD) or Python (PSF). However our license choice is compatible with these upstream licensing arrangements.

### 5.1.3 Privacy Notice

*see* Privacy Notice

### 5.1.4 Information Governance

### 5.1.5 Clinical Safety

### 5.1.6 Medical Device Registration

**Warning**

The textual portions of this work, documentation, and all other included non-code written information is licensed under the Creative Commons Attribution-ShareAlike 4.0 International license, except where otherwise stated.

# 5.2 Creative Commons Attribution-ShareAlike 4.0 International

Creative Commons Corporation (“Creative Commons”) is not a law firm and does not provide legal services or legal advice. Distribution of Creative Commons public licenses does not create a lawyer-client or other relationship. Creative Commons makes its licenses and related information available on an “as-is” basis. Creative Commons gives no warranties regarding its licenses, any material licensed under their terms and conditions, or any related information. Creative Commons disclaims all liability for damages resulting from their use to the fullest extent possible.

**Using Creative Commons Public Licenses**

Creative Commons public licenses provide a standard set of terms and conditions that creators and other rights holders may use to share original works of authorship and other material subject to copyright and certain other rights specified in the public license below. The following considerations are for informational purposes only, are not exhaustive, and do not form part of our licenses.

- **Considerations for licensors:** Our public licenses are intended for use by those authorized to give the public permission to use material in ways otherwise restricted by copyright and certain other rights. Our licenses are irrevocable. Licensors should read and understand the terms and conditions of the license they choose before applying it. Licensors should also secure all rights necessary before applying our licenses so that the public can reuse the material as expected. Licensors should clearly mark any material not subject to the license. This includes other CC-licensed material, or material used under an exception or limitation to copyright. More considerations for licensors.
- **Considerations for the public:** By using one of our public licenses, a licensor grants the public permission to use the licensed material under specified terms and conditions. If the licensor’s permission is not necessary for any reason–for example, because of any applicable exception or limitation to copyright–then that use is not regulated by the license. Our licenses grant only permissions under copyright and certain other rights that a licensor has authority to grant. Use of the licensed material may still be restricted for other reasons, including because others have copyright or other rights in the material. A licensor may make special requests, such as asking that all changes be marked or described. Although not required by our licenses, you are encouraged to respect those requests where reasonable. More considerations for the public.

## 5.2.1 Creative Commons Attribution-ShareAlike 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution-ShareAlike 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

**Section 1 – Definitions.**

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **BY-SA Compatible License** means a license listed at creativecommons.org/compatiblelicenses, approved by Creative Commons as essentially the equivalent of this Public License.

d. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

e. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

f. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

g. **License Elements** means the license attributes listed in the name of a Creative Commons Public License. The License Elements of this Public License are Attribution and ShareAlike.

h. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

i. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

j. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

k. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

l. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

m. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

**Section 2 – Scope.**

a. ***License grant.***

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

   A. reproduce and Share the Licensed Material, in whole or in part; and

   B. produce, reproduce, and Share Adapted Material.

2. **Exceptions and Limitations.** For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. **Term.** The term of this Public License is specified in Section 6(a).

4. **Media and formats; technical modifications allowed.** The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures. For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. **Downstream recipients.**

   A. **Offer from the Licensor – Licensed Material.** Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

   B. **Additional offer from the Licensor – Adapted Material.** Every recipient of Adapted Material from You automatically receives an offer from the Licensor to exercise the Licensed Rights in the Adapted Material under the conditions of the Adapter's License You apply.

   C. **No downstream restrictions.** You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. **No endorsement.** Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. ***Other rights.***

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

**Section 3 – License Conditions.**

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. ***Attribution.***

1. If You Share the Licensed Material (including in modified form), You must:

   A. retain the following if it is supplied by the Licensor with the Licensed Material:

```
 i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the
Licensor (including by pseudonym if designated);

 ii. a copyright notice;

 iii. a notice that refers to this Public License;

 iv. a notice that refers to the disclaimer of warranties;

 v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;
```

   B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

   C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.
3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

b. ***ShareAlike.***

In addition to the conditions in Section 3(a), if You Share Adapted Material You produce, the following conditions also apply.

1. The Adapter’s License You apply must be a Creative Commons license with the same License Elements, this version or later, or a BY-SA Compatible License.
2. You must include the text of, or the URI or hyperlink to, the Adapter's License You apply. You may satisfy this condition in any reasonable manner based on the medium, means, and context in which You Share Adapted Material.
3. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, Adapted Material that restrict exercise of the rights granted under the Adapter's License You apply.

**Section 4 – Sui Generis Database Rights.**

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material, including for purposes of Section 3(b); and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

**Section 5 – Disclaimer of Warranties and Limitation of Liability.**

a. **Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors, whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.**

b. **To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.**

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

#### Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or
2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

#### Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

#### Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the “Licensor.” The text of the Creative Commons public licenses is dedicated to the public domain under the CC0 Public Domain Dedication. Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at creativecommons.org/policies, Creative Commons does not authorize the use of the trademark “Creative Commons” or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at creativecommons.org.

## 5.3 Privacy Notice

The RCPCH has published a Privacy Notice for the Epilepsy 12 Project, this is available via the links below:

Epilepsy12 privacy notice for families 05/22 (English)

Epilepsy12 privacy notice for families 05/22 (Welsh)