RCPCH Epilepsy12 Documentation

User, Clinician, Admin and Developer Manual

The Royal College of Paediatrics and Child Health

Table of contents

1. Home 4
1.1 Introduction 4
1.2 KPIs and Quality Improvement 5
1.3 Citing the Epilepsy12 audit 6
2. Parent Guide 7
2.1 Parent's Guide 7
3. Clinician Guide 8
3.1 Overview 8
3.2 User Groups 9
3.3 Getting Started 10
3.4 Organisation View, Staff View and Cohort View 12
3.5 Creating, Editing and Deleting Users 13
3.6 Entering Patient Data 15
3.7 Audit Dataset 19
3.8 Organisational Audit 20
4. Administrator Guide 21
4.1 User permissions 21
4.2 Downloads 23
4.3 How to edit these docs 24
5. Developer Guide 27
5.1 Getting Started 27
5.2 Architecture Overview 29
5.3 Docker setup 31
5.4 Manual setup 38
5.5 Testing 41
5.6 Code Style 54
5.7 RCPCH Branding 55
5.8 Application Structure 56
5.9 API 90
5.10 Environment Variables 91
5.11 Deployment 92
5.12 Documentation 94
6. Contact 97
6.1 Contact Page 97

7. Legal 98
7.1 Intellectual Property 98
7.2 License (CC-BY-SA 4.0) 99
7.3 Clinical Safety 105
7.4 Privacy Overview 106
7.5 Privacy Notice 107
7.6 Data Protection Impact Assessment 108
7.7 Section 251 Exemption 109
7.8 Terms of Service 110

1. Home

1.1 Introduction

Guidance for Clinicians

Go to the Clinician Guide above for guidance on entering patient data.

1.1.1 Epilepsy12

Epilepsy12 (The UK collaborative clinical audit of health care for children and young people with suspected epileptic seizures) is a national clinical audit established in 2009 and has the continued aim of helping epilepsy services, and those who commission health services, to measure and improve the quality of care for children and young people with seizures and epilepsies. The audit is overseen by a project board and a dedicated project team within the RCPCH.

You can contact the team at: sitecontactemail@example.com or on 020 7092 6157/6056. Dr Colin Dunkley, Consultant Paediatrician, is the current clinical lead for Epilepsy12.

1.1.2 Childhood Epilepsy

Epilepsy affects around one in 200 children and young people in the UK (aged 18 and under). The condition is one of the most common significant long-term health conditions of childhood. Epilepsy can have a huge impact on children and families. Epileptic seizures can happen at any part of the brain cortex and what happens depends on where in the brain the seizure is happening. Seizures can be distressing but it's not just the seizures that are the problem. Children and young people may experience issues with learning, behaviour and emotions. That's why children and parents need support and guidance from healthcare professionals throughout the stages of diagnosis and treatment.

1.1.3 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.

1.2 KPIs and Quality Improvement

1.2.1 10 key performance indicators

1. Paediatrician with expertise in epilepsies
2. Epilepsy specialist nurse
3. a. Tertiary input b. Epilepsy surgery referral
4. ECG
5. MRI brain
6. Assessment of mental health issues
7. Mental health support
8. Sodium valproate
9. a. Care planning agreement b. Care planning content
10. School individual health care plan

1.2.2 Quality Improvement

Epilepsy12 engaged with relevant stakeholders to identify priority areas of care and agree five Health Improvement Goals to align with these.

1. Increase the proportion of children receiving input from an epilepsy specialist nurse by 5% per year; from 76% in Cohort 3 to 91% in Cohort 6.
2. For children with complex epilepsy, increase the proportion receiving input from a tertiary specialist by 5% per year; from 69% in Cohort 3 to 84% in Cohort 6.
3. Increase the proportion of children receiving all core elements of care planning by 5% per year; from 75% in Cohort 3 to 90% in Cohort 6.
4. Increase the proportion of Health Boards and Trusts using structured transition resources by 5% per year; from 62% in Cohort 3 to 77% in Cohort 6.
5. Increase the number of Health Boards and Trusts screening children with epilepsy for mental health disorders by 5% per year; from 19% in Cohort 3 to 34% in Cohort 6.

Epilepsy12 collaborated with QI experts when designing the improvement goals and strategies. We will continue to receive their input when delivering these strategies and seek guidance on how we can embed Epilepsy12's evaluation and learning to improve the plan and explore avenues of collaboration wth NHS, HQIP, and other improvement programmes.

You can find out more about our quality improvement plans for 2022-25 in our Quality Improvement Strategy.

1.3 Citing the Epilepsy12 audit

1.3.1 Citation of the Epilepsy12 audit in academic publications

You can use our Zenodo DOI button to cite this audit

DOI 10.5281/zenodo.6549072

2. Parent Guide

2.1 Parent's Guide

2.1.1 Childhood Epilepsy

Epilepsy affects around one in every 200 children and young people in the UK (aged 18 and under). The condition is one of the most common significant long-term health conditions of childhood.

Epileptic seizures can happen in any part of the brain cortex and what happens depends on where in the brain the seizure is happening. Seizures can be distressing but it's not just the seizures that are the problem. Children and young people may experience issues with learning, behaviour and emotions.

Epilepsy can have a huge impact on children and families. That's why children and parents need support and guidance from healthcare professionals throughout the stages of diagnosis and treatment.

3. Clinician Guide

3.1 Overview

You can find more information about the Epilepsy12 audit, including previous reports, on our webpage.

3.1.1 Overview

The Epilepsy12 audit is delivered by the Royal College of Paediatric and Child Health (RCPCH), who have been commissioned by the Healthcare Quality Improvement Partnership (HQIP), as part of the National Clinical Audit and Patient Outcomes Programme (NCAPOP). Round 4 is underway and is being delivered from 1 April 2022 to 31 March 2025.

Patients are grouped into cohorts based on the date of their first paediatric assessment. We collect the data on the first year of care of each patient following the first assessment.

Cohort	Date of first paediatric assessment	First year of care data entry timeframe	Data entry deadline
5	01 December 2021 - 30 November 2022	01 December 2022 - 30 November 2023	09 January 2024
6	01 December 2022 - 30 November 2023	01 December 2023 - 30 November 2024	14 January 2025
7	01 December 2023 - 30 November 2024	01 December 2024 - 30 November 2025	13 January 2026

3.2 User Groups

3.2.1 User Groups

There are three user groups for staff working in Trusts/Health Boards, with varying permissions.

Lead Clinician

- Can create, edit and delete users within their Trust/Health Board
- Can create, edit, and delete patients within their Trust/Health Board
- Can create, edit, and delete patient audit records

Clinician

- Can view users within their Trust/Health Board
- Can create, edit and delete patients within their Trust/Health Boards
- Can create, edit and delete patient audit records

Administrator

- Can view users within their Trust/Health Board
- Can create and edit patients within their Trust/Health Board
- Can view patient audit records

These user groups are linked to one or multiple Trust/Health Boards which are providing epilepsy care to children and/or young people. Users are assigned to an 'organisation', which is defined as individual hospitals/sites/community services within one Trust/Health Board.

Users will only have access to the data relating to their own Trust/Health Board. They may see children and users at other organisations within their Trust/Health Board.

3.3 Getting Started

3.3.1 Getting Started

Making a new account

Lead Clinicians from Trusts/Health Boards will need to contact the Epilepsy12 team to create their account.

Clinicians and Administrators can ask their Lead to create an account for them. Alternatively, they can contact the Epilepsy12 team, who can create their account with written permission from the relevant Trust/Health Board Lead Clinician.

If you are a lead, you can make new accounts by navigating to the `Staff View`, selecting the relevant organisation, and clicking the 'Add new user' button.

If your Trust/Health Board does not have a lead registered to the audit already, you will need select a Lead clinician and we will require approval from the Trust/Health Board's Caldicott Guardian. This is to ensure that we are only providing trusted individuals with access to the Epilepsy12 platform and the data hosted within the system.

When your account is created, you'll be emailed a link to set your password. This link expires after **72 hours**. If your link has expired, please contact the Epilepsy12 team to send you a new link.

Logging in

Click the 'Sign in' button on the home page to log in.

The first time you log in, you will need to set up 2 factor authentication. This is an additional security measure to prevent unauthorised access to patient data. You will have the choice to either set up an email token or an authenticator app.

EMAIL TOKEN

If you select email token, you will receive a six digit code to your inbox. You will need to enter that code to log in.

AUTHENTICATOR APP

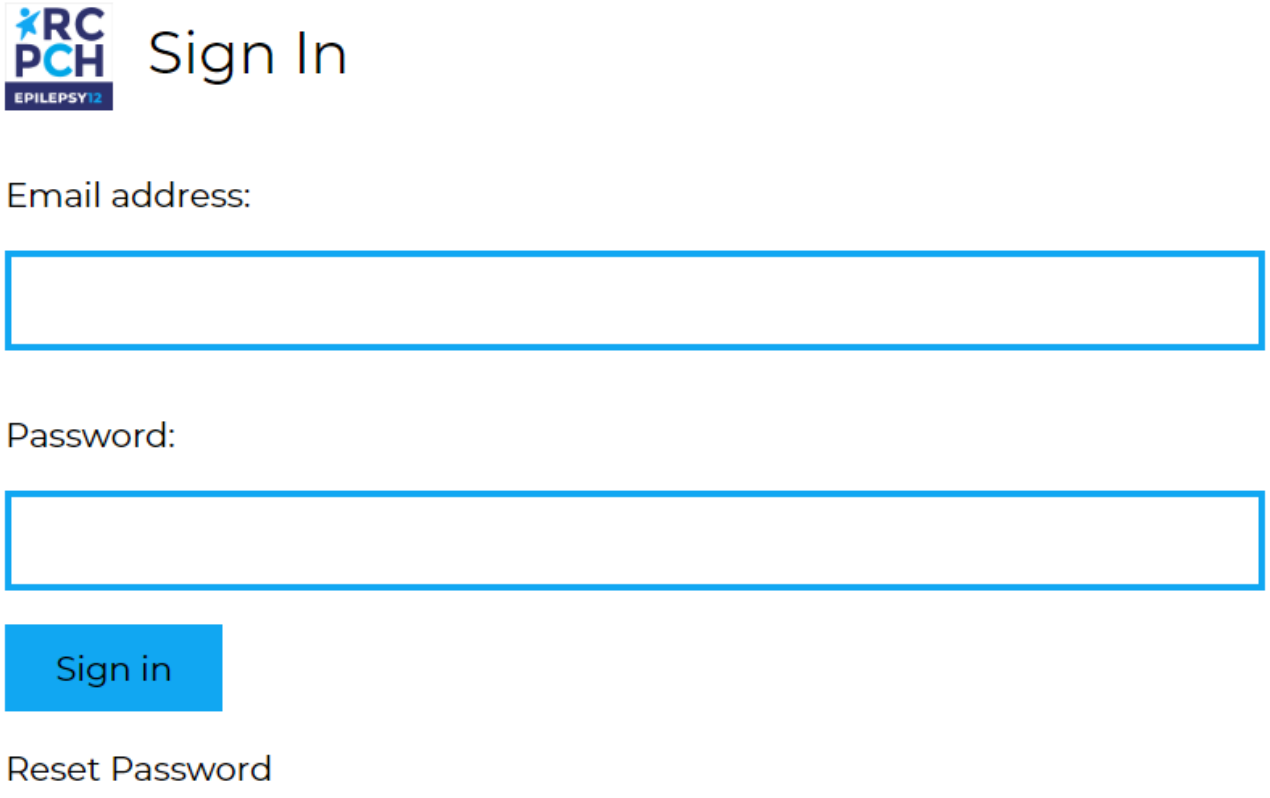
If you select authenticator app, you will need to download the Microsoft Authenticator app on a smartphone or tablet. Once you have downloaded and logged into the app, you will need to follow the instructions provided to set up the authenticator app.

Each time you log in, you will need to access your authenticator app and enter your one-time password to log in. This password will be a six digit code that resets every 30 seconds.

If you lose access to your authenticator app, please contact the Epilepsy12 team.

Reset password

If you have forgotten your password, click 'Reset Password' and a link to create a new password will be sent to your email address, and this will remain active for 72 hours.



3.4 Organisation View, Staff View and Cohort View

3.4.1 Organisation View, Staff View and Cohort View

Organisation View

When you first log in, you'll be taken to the **Organisation View**. Here, you'll see information about your organisation, the current audit cohort, and real-time summary data for key performance indicator metrics.

Your organisation details are automatically pulled from the Organisation Data Service. If you are an Lead Clinician, you can navigate between the organisations within your Trust/Health Board.

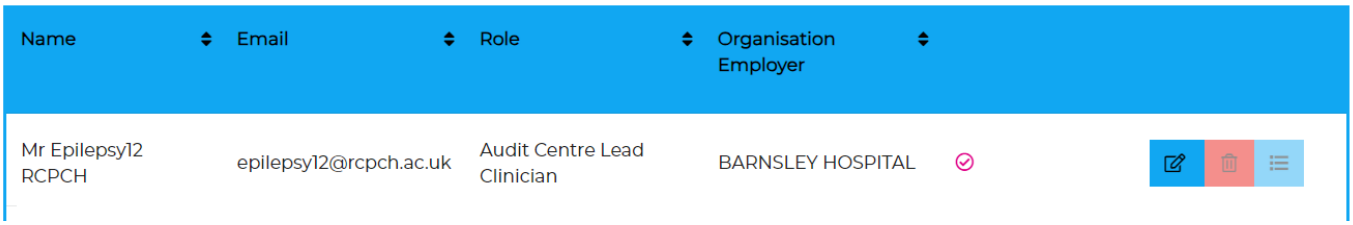
The real-time **Key Performance Indicator (KPI) dashboard** is a live report that updates every time a patient's record is completed. This report allows you to compare your organisation's performance with your Trust/Health Board, Integrated Care Board (England), NHS Region (England), OPEN UK Region, country, and England & Wales combined.

If you go into the **Individual Measures tab**, you can see visual comparisons of a single performance indicator across all ICBs, NHS Regions, OPEN UK Regions, and countries.

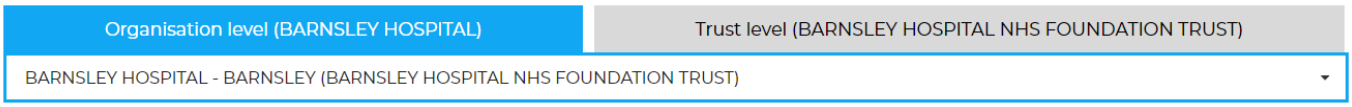
Staff View

To enter the Staff View, click the '**View All Epilepsy12 Staff**' button in the Organisation View.

Here you will see a list of all the platform user accounts for your Trust/Health Board. You will see each user's name, email address, role, and organisation. A pink tick indicates that the user is active.



You can navigate between organisations in the same Trust/Health Board using the dropdown menu.



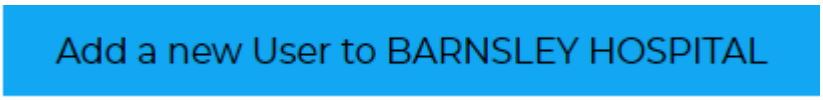
3.5 Creating, Editing and Deleting Users

3.5.1 Creating, Editing and Deleting Users

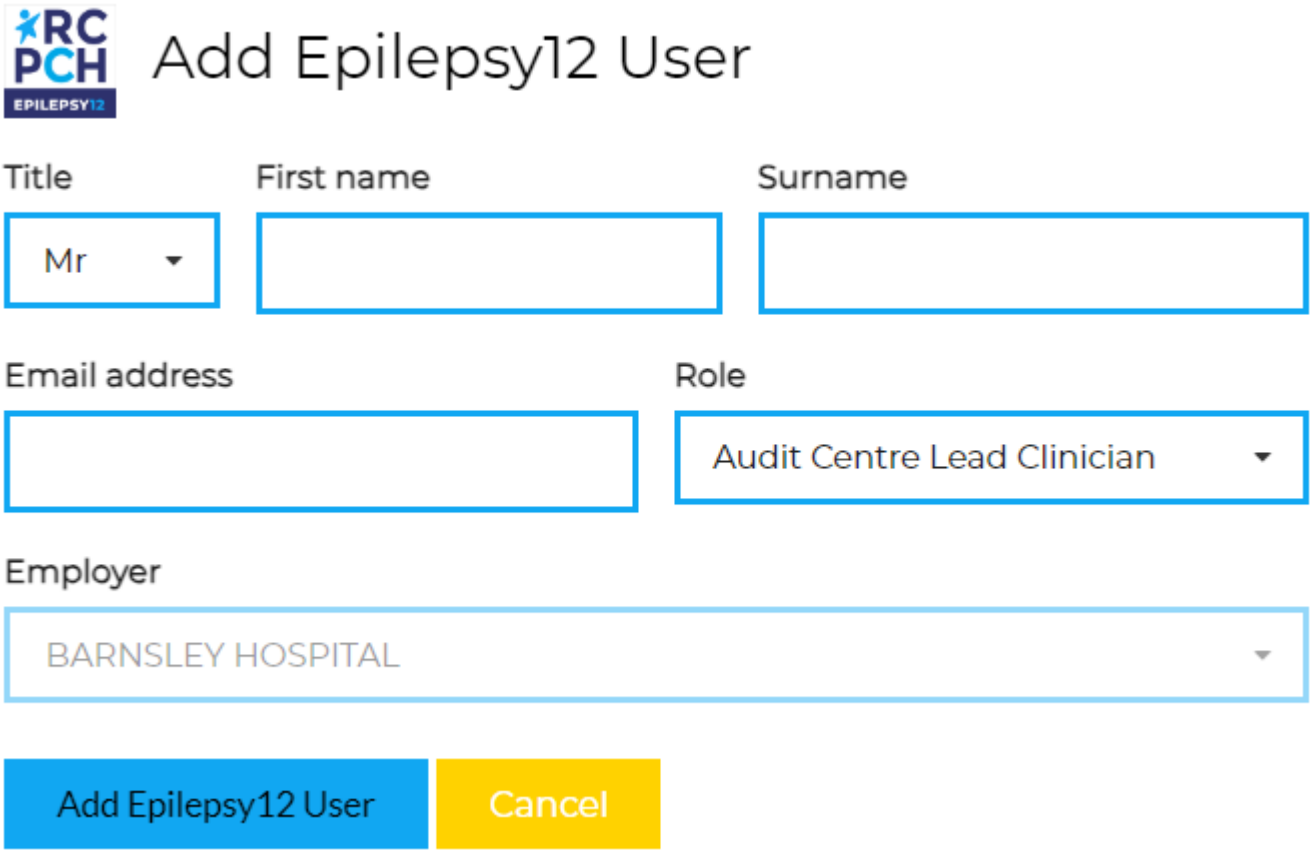
Only lead clinician users can create, edit and delete users.

Add a new user

To add a new user, navigate to the **staff view** and make sure you have selected the correct organisation at the top of the screen. Then click on the **Add a new user to ORGANISATION** button.

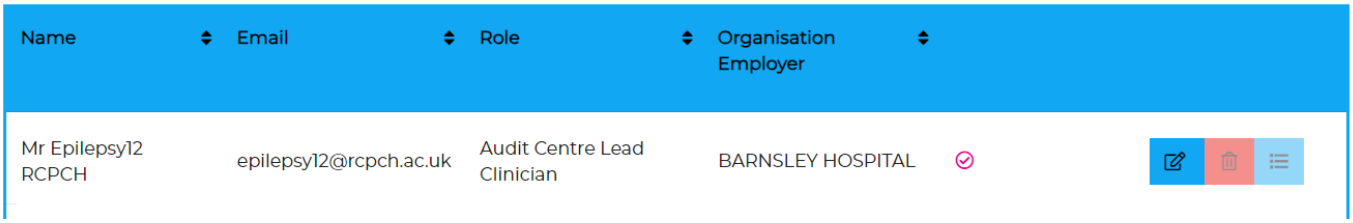


You will need to provide the user's title, full name, email address and user group.

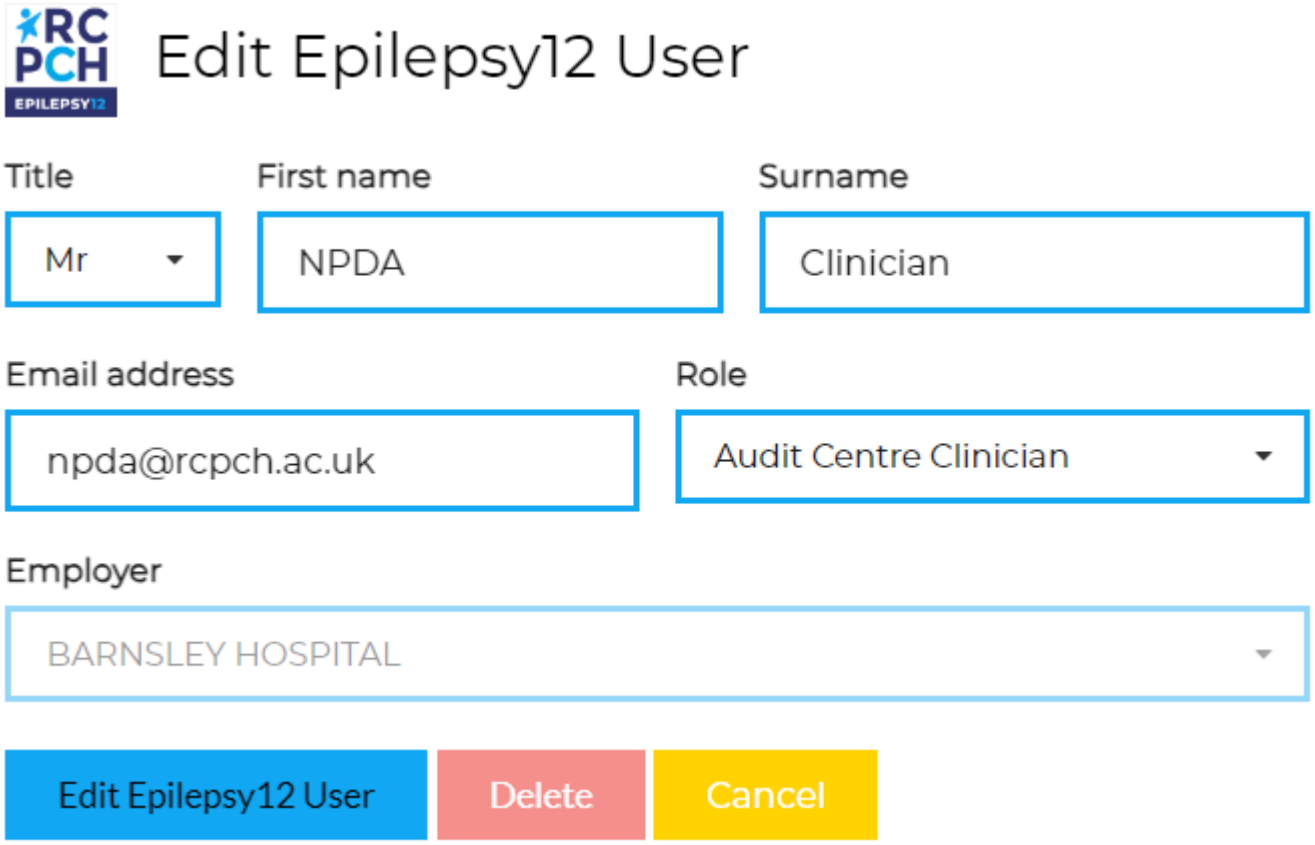


Editing and deleting users

To edit or delete a user, click the edit icon.



You will then be able to edit the user's details or delete the user by clicking the red delete button.



3.6 Entering Patient Data

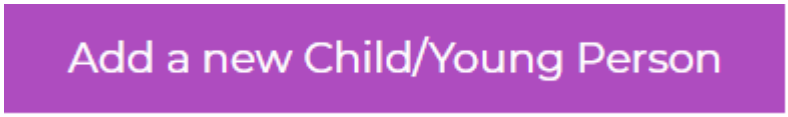
3.6.1 Entering Patient Data

Consent and opt out

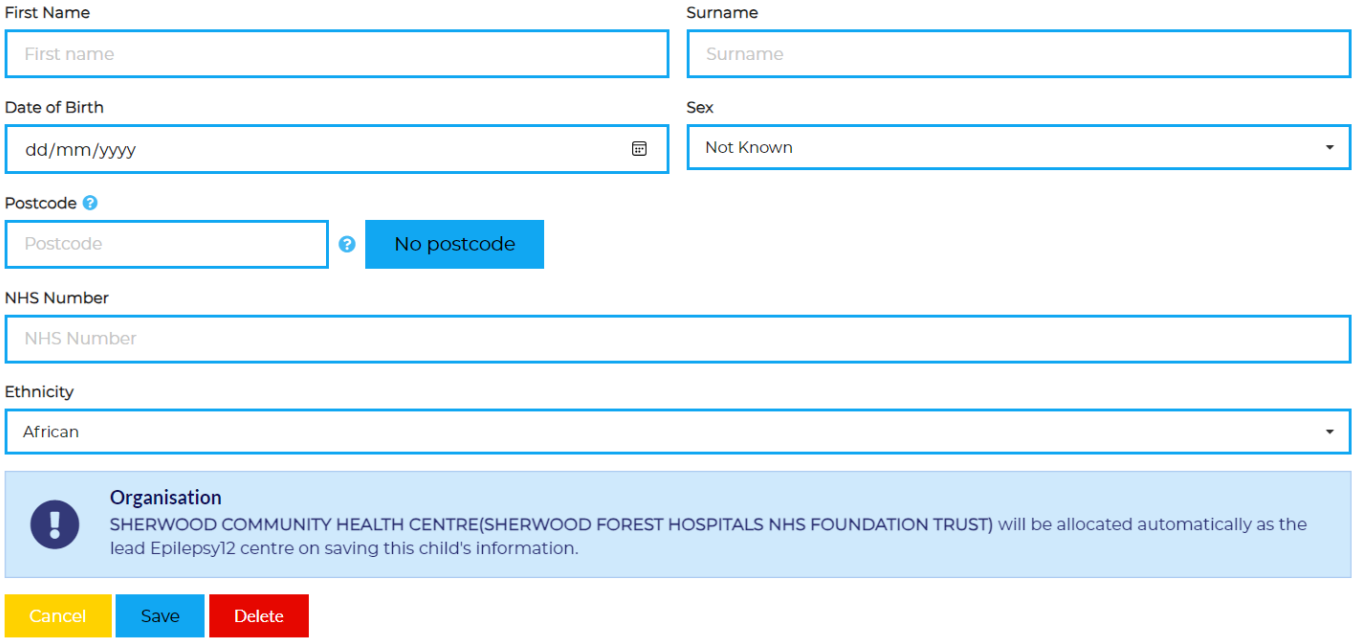
Please note that Epilepsy12 has received an exemption the NHS National Data Opt-out (England only) and all eligible patients can therefore be registered onto the platform regardless of their NDO status. Children and young people or their guardians may still choose not to participate in the Epilepsy12 audit specifically. If you add a patient to the audit and they choose to opt-out, you can delete their record. View the guidance below on this.

Add a new patient

To add a new patient, navigate to the **cohort view** and click the **Add a new Child/Young Person** button



You will need to provide the patient's first name, surname, date if birth, sex, postcode, NHS number and ethnicity.



This is not real patient data

If you are unable to provide a postcode, click the **No postcode** button and you will be asked to indicate the reason.

The NHS number is checked against the NHS number checksum to confirm that it is a valid NHS number. We also use this to check if the patient has already been entered into the audit at a previous date, to avoid duplication.

Transferring a patient between centres

Only the **Lead Clinician** has permissions to transfer children to another centre.

The steps to do this for the lead clinician in Organisation A

1. Find the child to transfer in the cohort view and navigate to their audit view
2. Select the 'transfer' button to reveal a dropdown of organisations to transfer the child to.
3. Select the organisation to transfer the child to (Organisation B) and click 'Select Secondary Care NHS Trust Centre'
4. Confirm the transfer by clicking 'Allocate organisation'
5. You will be redirected back to the organisation view, where a message box will confirm a transfer email has been sent to the lead clinician of the target organisation (Organisation B). Note if there is no lead clinician currently in post, the email will be sent to the Epilepsy12 team.
6. The child will no longer be visible to you in Cohort View for Organisation A.

The steps for the lead clinician in Organisation B

1. You receive an email from your colleague who leads Epilepsy12 at Organisation A, notifying you that a transfer has been made.
2. Log in to Epilepsy12. You are greeted with a popup message informing you of the name of the child requesting transfer.
3. Navigate to the Cohort View and identify the child - they are highlighted in the list and have 2 extra icons.
4. Click whichever applies (accept or reject), and acknowledge the confirmation popup.
5. You will be redirected to the organisation view where a confirmation message will acknowledge successful outcome.
6. An email is sent to the lead of Organisation A is sent to notify the outcome. Note, if the transfer is rejected, an email also is sent to the Epilepsy12 team.

Entering audit data

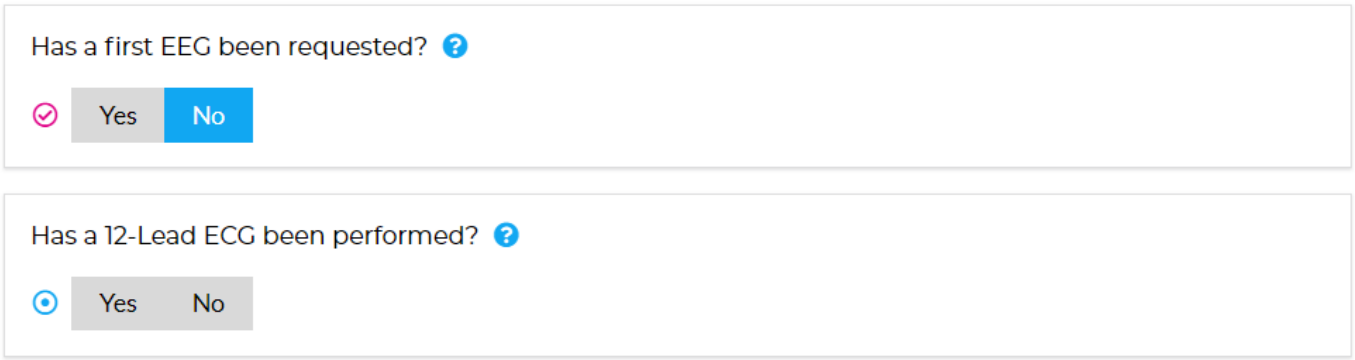
Only clinician and lead clinician users will be able to do this.

Once a patient has been added to the audit, you can navigate to the patient’s record in the cohort View. Click on the **Audit** button to the right of their name to access the audit data.

You will need to verify that the patient meets the eligibility criteria for the audit before you can enter any data. Please note that this step is **not reversible**, so please read through the criteria carefully before confirming eligibility. For more information on this, please refer to our methodology overview.

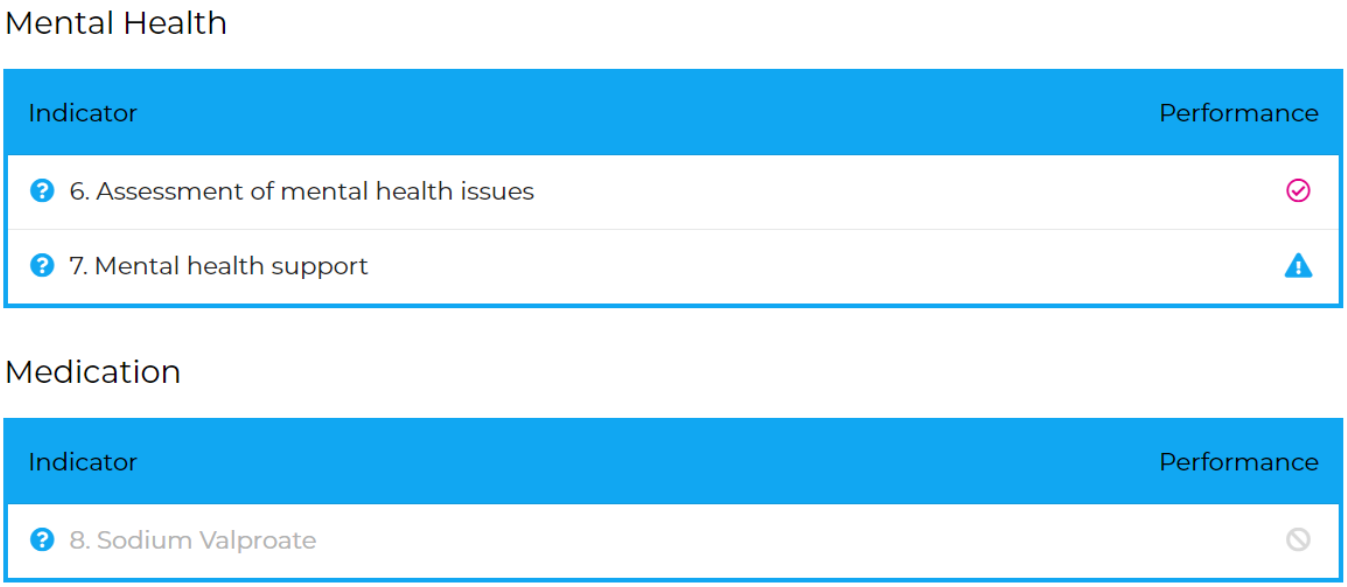
You can navigate through pages in the audit form using the navigation pane on the left. Please complete all the questions in each section to reflect the care provided to the patient during the **first 12 months of care** following a first paediatric assessment. Each tab will turn pink once completed.

A blue dot next to a question indicates that it is incomplete. A pink tick will appear once the question is complete.



Performance Summary

This is the final tab of the audit form. Here you can see if the patient has successfully met each of the key performance indicator metrics. A pink tick signals they have met the indicator already and a blue triangle means that they have not yet met the indicator. A grey stop sign means that this performance indicator is not applicable for this patient, and this may appear if the patient does not meet the criteria for inclusion within the metric.



Delete a patient

Only lead clinician and clinician users can delete patient records. If a patient was erroneously added to the audit, or has opted-out, you can delete their record.

Enter the cohort view and find the patient's record. Click the **edit** button, which will take you to the patient details page. Scroll down and click the **Delete** button.

3.7 Audit Dataset

3.7.1 Audit Dataset

The Key Performance Indicators for Round 4 of the Epilepsy12 audit are outlined below. For more information on the Round 4 performance indicators, including the relevant guidelines and submetrics, please see our Round 4 methodology overview.

3.8 Organisational Audit

3.8.1 Organisational Audit

The Organisational Audit collects data on the paediatric epilepsy services within each Trust/Health Board. For the 2023 Organisational Audit, please complete this on the old Epilepsy12 platform.

4. Administrator Guide

4.1 User permissions

These are the permissions held by each User Group.

User Group	Scope	Permission Target	Permissions
RCPCH Audit Team / RCPCH Clinical Audit Team	National	E12 User	Create, View, Update, Delete
	National	E12 Patients	Create, View, Update, Delete, Transfer
	National	E12 Patient Records	Create, View, Update, Delete
Lead Clinician	Trust	E12 User	Create, View, Update, Delete
	Trust	E12 Patients	Create, View, Update, Delete, Transfer
	Trust	E12 Patient Records	Create, View, Update, Delete
Clinician	Trust	E12 User	View
	Trust	E12 Patients	Create, View, Update, Delete
	Trust	E12 Patient Records	Create, View, Update, Delete
Administrator	Trust	E12 User	View
	Trust	E12 Patients	Create, View, Update
	Trust	E12 Patient Records	View

4.1.1 User types and permission detail

RCPCH Audit Children and Family

- Can **view** their own data.
- Can consent to participation. Can remove consent. Can opt out. Opting out leads to all data related to them being **delete**d, except the Epilepsy12 Unique Identifier.

Audit Centre Lead Clinician

- Can delete all data relating to their **trust(s)**.
- Can view, add, edit and delete a child's demographic, case and clinical information.
- Can sanction a child opting out of Epilepsy12
- Can lock and unlock a child's record from editing.

- Can approve eligibility for Epilepsy12 and remove approval of eligibility.
- Can register and unregister a child in Epilepsy12
- Can view, allocate, update and delete and child's general paediatric centre or tertiary neurology centre.
- Can request transfer for a child to a different Epilepsy12

Audit Centre Clinician

- Can edit but not delete all data relating to their **hospital(s)**.
- Can view, add, edit and delete a child's demographic, case and clinical information.
- Can sanction a child opting out of Epilepsy12
- Can lock and unlock a child's record from editing.
- Can approve eligibility for Epilepsy12, but not remove approval of eligibility.
- Can register a child in Epilepsy12, but not unregister.
- Can view, allocate, update and delete a child's general paediatric centre or tertiary neurology centre.

Audit Centre Administrator

- Can view, but not edit or delete, data relating to their **hospital(s)**.
- Can add, view, edit but not delete, a child's demographic and case information, but not clinical.
- Can view, but not edit or delete, a child's general paediatric centre or tertiary neurology centre.

RCPCH Team

- Can create, update, and delete **national** data, logs, epilepsy key words and hospital trusts, groups and permissions.
- Can lock and unlock a child's record from editing.
- Can sanction a child opting out of Epilepsy12
- Can view and edit a child's demographic, case and clinical information.
- Can approve eligibility for Epilepsy12 and remove approval of eligibility.
- Can register and unregister child in Epilepsy12.
- Can view, allocate, update and delete a child's general paediatric centre or tertiary neurology centre.
- Can publish data in the dashboards to the public domain

4.2 Downloads

4.2.1 PDF export

You can download the entire Epilepsy12 documentation manual as a PDF for offline reading. Click the button below to download.

Download documentation manual in PDF format

4.3 How to edit these docs

4.3.1 Markdown Files

This site is created using the MkDocs documentation framework. All text content comes from markdown files.

Markdown is a simple, lightweight markup language that allows you to format text using a few simple symbols. It was designed to be easy to read and write, even for people unfamiliar with coding.

In Markdown, you can format text by using a combination of characters such as `*` (asterisks), `_` (underscores), and `-` (dashes) to create headings, bold and italic text, lists, links, and more.

For example, to create a heading, you start the line with one or more `#` symbols, followed by the title of your heading.

For example:

```
# This is a level 1 heading

## This is a level 2 heading

### This is a level 3 heading
```

To create a list, you can use a `*` or `-` symbol followed by the item you want to list. For example:

```
- Item 1
- Item 2
- Item 3
```

You can also create ordered lists by using numbers:

```
1. First item
2. Second item
3. Third item
```

Markdown also allows you to add links and images to your text using a simple syntax:

```
[Link text](URL)

![Alt text](image URL)
```

Markdown is widely used on the web, especially in blogs, documentation, and forums. It's a great way to format your text without learning complex HTML or other markup languages.

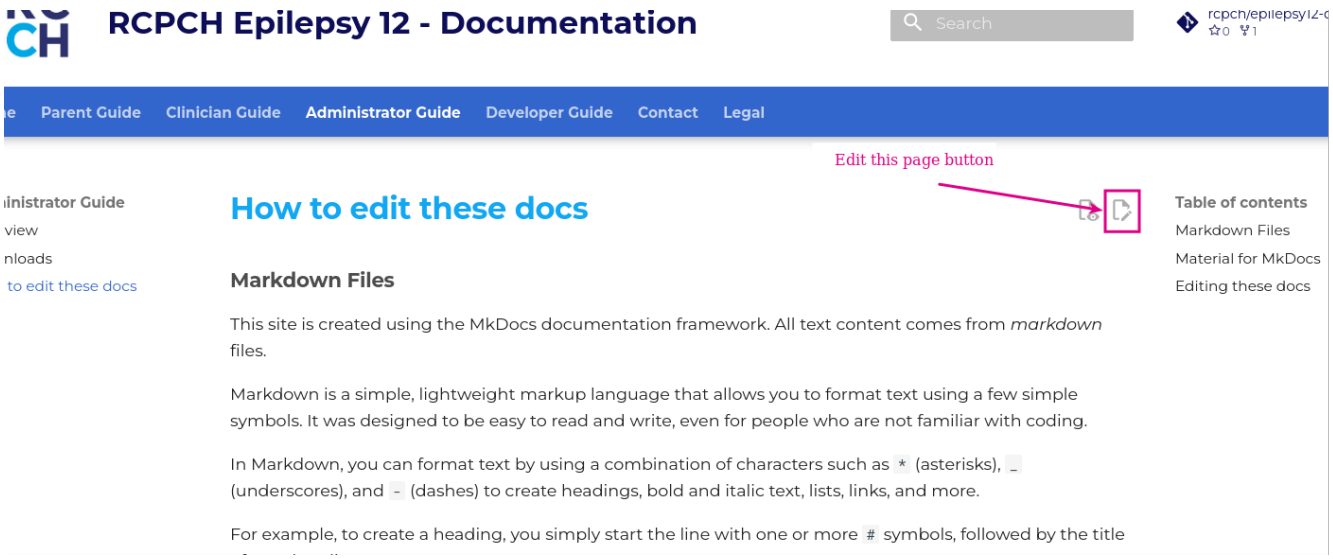
4.3.2 Material for MkDocs

On top of MkDocs, this site uses the 'Material for MkDocs' theme, which adds several extra features and a more modern appearance. In addition, we use the Material for MkDocs Insiders edition, allowing us to support the project whilst getting a few neat early-access features.

As you'd expect, there is delightful documentation for both projects: Material for MkDocs, and for the underlying MkDocs, on which it's built. At times, you may need to refer to both for different features.

4.3.3 Editing these docs

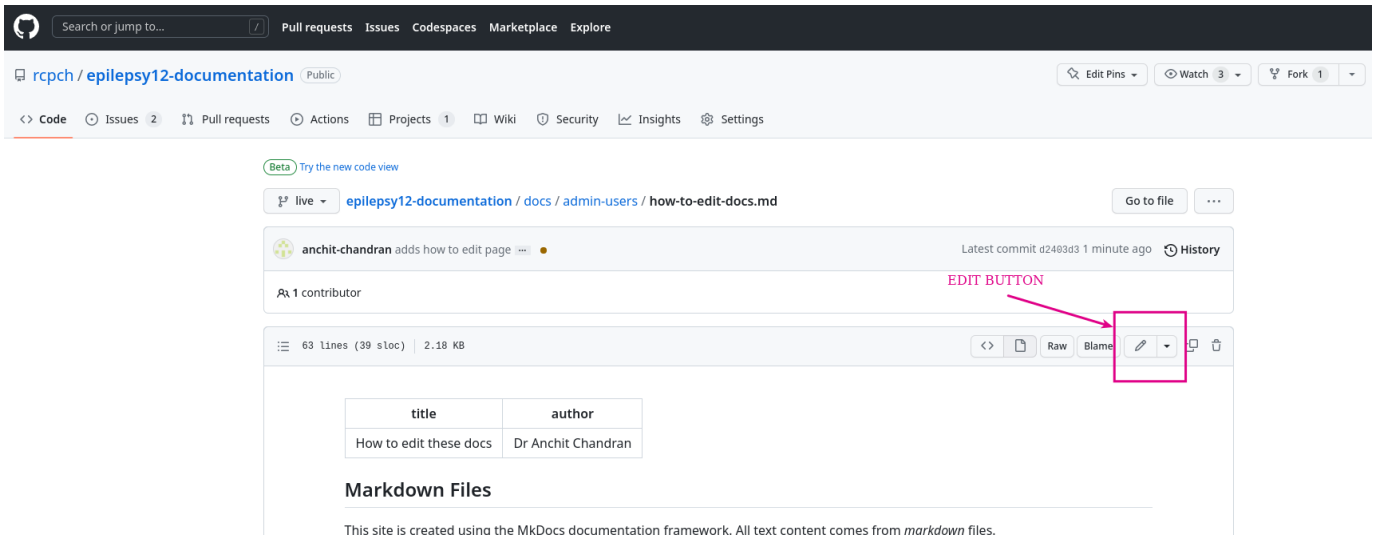
The simplest way is clicking on the '`Edit this page`' button at the top right of the page you wish to edit:



You will be taken to the edit page on GitHub.

GitHub Edit Page

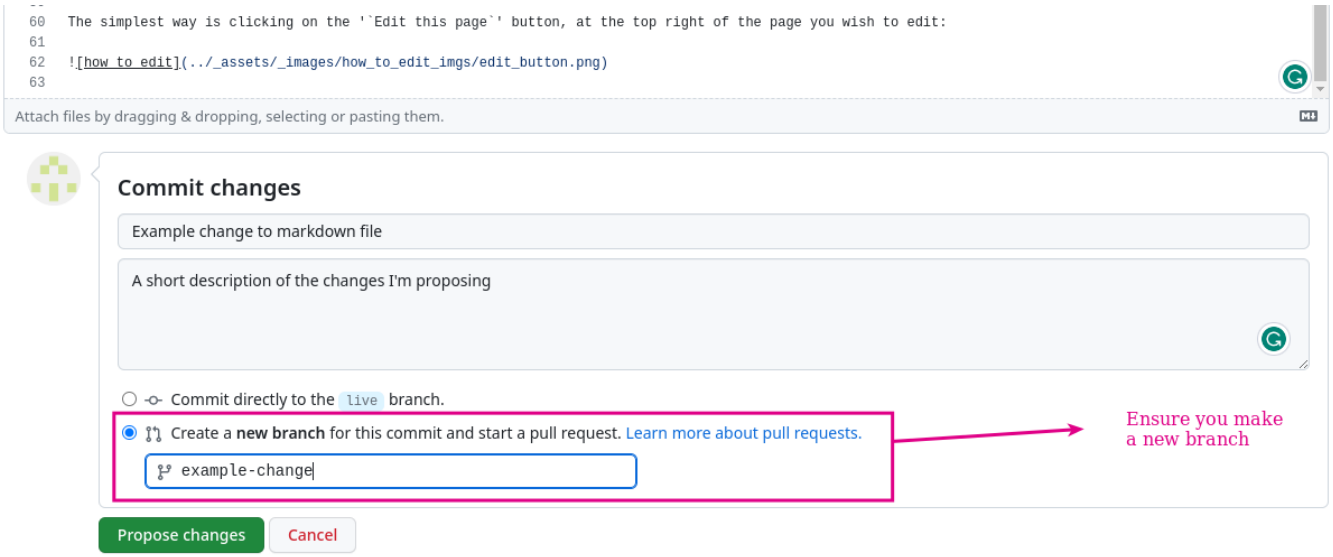
Again, click on the edit button near the top right:



Make your edits

Make any changes you wish to suggest on this page. You can see a preview of the output markdown using the 'Preview' button (note: Material for MkDocs' flavour of markdown is slightly different to GitHub's, so it may not appear perfectly.)

Once done, scroll to the bottom of the file. Here, you will submit the proposal for your change:



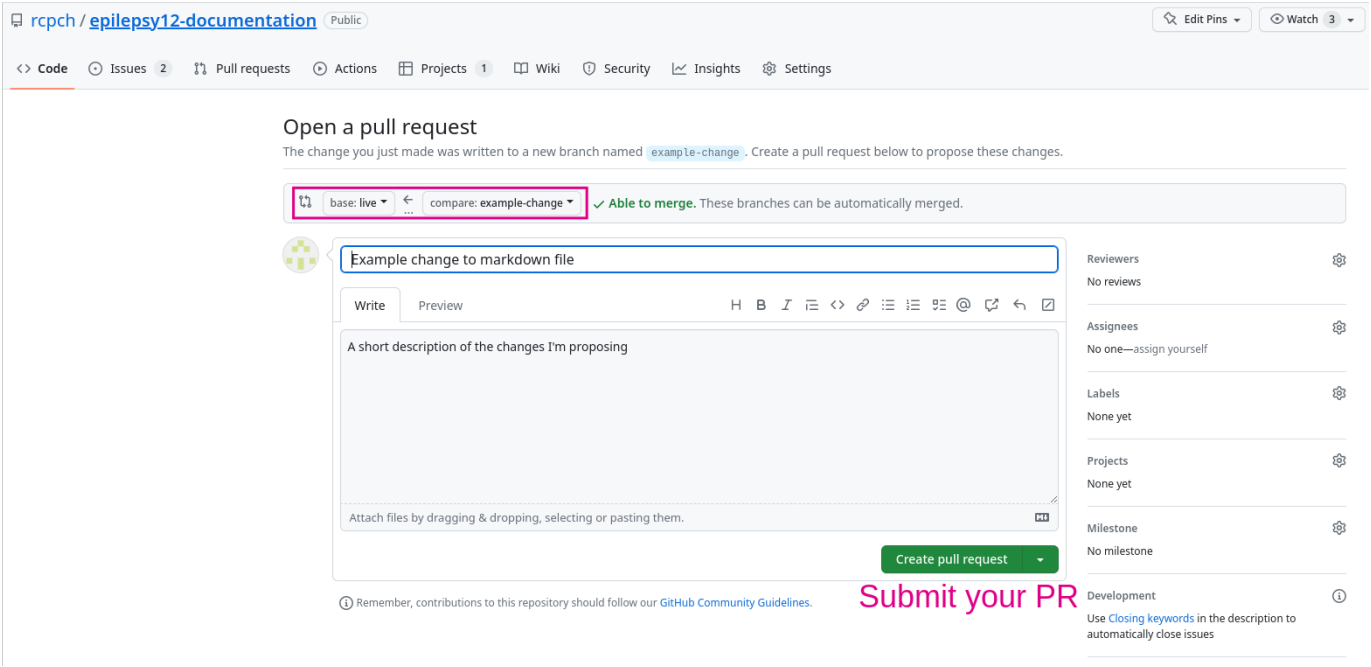
- Enter a suitable title and description for your proposal, so one of the developers can quickly understand its purpose.

> **Make a new branch**
>
> Make sure you select **"Create a new branch for this commit and start a pull request."**

Submit your Pull Request

You'll be taken to the 'Submit a Pull Request' page. Review the details of your request.

Once everything looks good, submit your change proposal by using the green `'Create pull request'` button:



5. Developer Guide

5.1 Getting Started

5.1.1 Introduction

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. Each audit is likely to have some common features and some bespoke features, but the audit engine aims to make the common features easier to build.

5.1.2 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, providing structure, security features, and numerous prebuilt features and functionality which save time and reduce errors when developing.

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 online course. Many other similar courses are available, often for free online. Django itself has a great tutorial here.

5.1.3 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.

5.1.4 Open Source

The entire output of the RCPCH Incubator development team is open source. We believe in the importance of open source 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

5.1.5 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.

5.1.6 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.

5.1.7 Legal

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

5.2 Architecture Overview

5.2.1 Epilepsy12 Application Architectural Overview

Introduction

This overview was originally prepared in order to assist the Privacy Impact Assessment of the E12 platform, but it is of general utility and interest, so has been made part of the documentation site for the project.

Cloud services

All of the Epilepsy12 application services run in Microsoft Azure cloud services platform, with all the individual resources chosen to be in the UK South data centre region.

Access to the cloud services platform is controlled with named accounts throughout, all of which use two-factor authentication in addition to a username and password combination. Accounts are managed by the RCPCH IT team, and access is granted on a named-individual basis, with limitation of scope of that user's access to the services they would need to manage to do their work.

Epilepsy12 Django Application

The main application is a web application, written in the **Python** language and using the **Django** framework. It is currently served using Azure App Services which is a cloud-based application deployment platform, but in the future we are transitioning to use a more 'traditional' approach, deploying the application and all dependencies to a virtual machine in the cloud.

A secure connection (HTTPS, also known as SSL) is used to access the application from the web, which allows secure logins and prevents private data being intercepted in transit.

Database

The application uses a **PostgreSQL** database, which is a common and reliable open-source relational database management system used by millions of other applications. The database is part of the same docker compose setup as the Django app itself and runs on the same VPS. This database is only used for the Epilepsy12 application, and is not shared with any other application.

All the data in the live instance of the application is stored in the database. The VPS running the app and database is backed up daily.

Authentication

Users authenticate with the application using an email address, and a password. The password is stored in the database, but is encrypted using a secure hashing algorithm, which performs a kind of one-way mathematical function on the password data. This means that even if the database is compromised, the passwords cannot be discovered. However the application can still check that someone has entered the right password by comparing the hash of the password they entered with the hash stored in the database.

Accounts are created manually by RCPCH admin or organisation lead clinicians who first verify that the proposed new user has the right to access the application. The application does not allow users to create their own accounts, since our user base is very specifically defined as the contributors to the Epilepsy12 audit.

WORKFLOW

The lead clinician or RCPCH admin create an account in the user management area. This creates an account with the email_confirmed flag set to false, and generates an email to the new user with an individualised and hashed token in the URL. This remains valid for 72 hours, after which time admin can send a new email if the user requests one by emailing the admin team or lead clinician. The email link redirects the user to reset a password which on creation sets the email_confirmed flag to

true. This is shown in the user management platform as a pink tick, along with any other badges to denote their status (RCPCH team, superuser etc).

PASSWORD RULES

Passwords must be 10 characters long for regular users, 16 characters for superusers or RCPCH admin. Passwords must contain 1 capital, 1 number and 1 symbol. Passwords are valid for 3 months, after which the user is asked to reset them. All login attempts are logged and if a user fails to log in 5 consecutive times, they are locked out of the platform for 10 minutes.

Authorisation

User accounts are limited to a single role, which determines what they can do within the application. The exact roles and capabilities are detailed in the admin user guide.

Access constraints

- Users at a specific clinical site can only access data for children at that Trust.
- Users with a national role can access data for all children.

5.3 Docker setup

5.3.1 Docker setup

To simplify the development environment setup and provide greater consistency between development and production environments, the application is built as a Docker image and Docker Compose sets up the other necessary containers in development.

This means you don't need to worry about conflicts of Python versions, Python library versions, or Python virtual environments. Everything is specified and isolated inside the Docker containers.

Setup for development using Docker Compose

INSTALL DOCKER ON YOUR DEVELOPMENT MACHINE

Instructions for all platforms are at get-docker

CLONE THE AUDIT ENGINE REPOSITORY TO YOUR CODE FOLDER

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

NAVIGATE INTO THE FOLDER

```
cd rcpch-audit-engine
```

> **Windows Setup**
>
> **If you are on Windows**, after installing Docker and cloning the repository, please now skip to the (Windows) Setup for development using Docker Compose section.

ENSURE YOU ARE ON THE DEFAULT `DEVELOPMENT` BRANCH

```
git checkout development
```

OBTAIN A `.ENV` FILE CONTAINING THE REQUIRED ENVIRONMENT FILES

These files contain credentials and secrets and therefore the `.env` files themselves are **never** committed to version control. All `*.env` files are `.gitignore`'d, as is the entire contents of the `envs/` folder except the env_template file.

If you work with the RCPCH Incubator team, another member of the team may be able to supply you with a completed `.env` file.

For anyone else, there is a template environment file in the repository root which you can rename to `.env` and use as a starting point. **Be extremely careful to make sure it is named `.env` so that it is ignored by Git. Do not ever commit `.env` files to version control!**

```
cp envs/env-template envs/.env
```

> **Mac Users**
>
> If using Mac and Safari, to access the Epilepsy 12 engine in your development, you must change the `SITE_DOMAIN` name in .env to 'localhost', and type this into your browser once you have executed `s/up` in the next step. This will load the E12 engine in your Safari browser.
>
> However, for simplicity, we recommend using a different browser, such as Chrome, and leaving the .env file unaltered.

START THE DEVELOPMENT ENVIRONMENT FOR THE FIRST TIME USING OUR STARTUP SCRIPT

```
s/up
```

This script automates all the setup steps including upgrading `pip`, installing all development dependencies with `pip install`, migrating the database and seeding the database with some essential data.

TRUST THE CADDY ROOT CA CERTIFICATE

To get a HTTPS connection through the Caddy server to work on `e12.localhost`, you need to trust the Caddy root CA certificate. This is only necessary once. We have a script to automate this.

```
s/trust-caddy-root-ca
```

If you encounter problems, further instructions for this are in the Caddy documentation. With some browsers you may need to manually add the certificate in the Security settings, see the above link for details. Restart the browser after trusting the certificate.

STARTUP

`s/up` will build the necessary Docker images, create the containers, and start them up. There will be a lot of output in the terminal, but it should create a number of containers and network them together. If you hit errors, please do open an issue.

At the very end of the terminal output, which could take several minutes, you should see something like this:

```
rcpch-audit-engine-web-1          | Django version 4.2.5, using settings 'rcpch-audit-engine.settings'
rcpch-audit-engine-web-1          | Starting development server at http://0.0.0.0:8000/
rcpch-audit-engine-web-1          | Quit the server with CONTROL-C.
```

> **https://e12.localhost, not http://localhost:8000**
>
> **IMPORTANT: Because we are using the Caddy web server as a reverse proxy, the application should be accessed at https://e12.localhost, not http://localhost:8000, even though Django will still report that is the hostname and port it 'thinks' it is listening on.**

Changes you make in your development folder are **automatically synced to inside the Docker container**, and will show up in the application right away, as long as your .env file is configured with `DJANGO_STARTUP_COMMAND="python manage.py runserver 0.0.0.0:8000"`. (We have other startup commands we use in production environments which don't have auto-reload)

Please do open an issue if there is anything that doesn't seem to work properly. Suggestions and feature requests welcome.

WHAT DOES `S/UP` DO?

This script automates all the setup steps including:

- upgrading `pip`
- installing all development dependencies with `pip install`
- migrating the database
- seeding the database

The `django` container is built with the correct Python version, all development dependencies are automatically installed, the database connection is created, migrations applied and some seed data is added to the database.

View the application in a browser at .

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.

> ⚠ **Terminal is now occupied**
>
> If you have successfully run the Docker Compose deployment, your terminal will be showing the combined and colour-coded logging output for all the containers and will no longer show an interactive prompt, which is means you can not run any more commands in that terminal. To resolve this, simply **open another Terminal window** in the same working directory, in which you can run commands.
>
> If opening another terminal is impractical or impossible, then in most Shell environments you can press `Ctrl` + `Z` to suspend the current process, and then `bg` to resume it in the background. This will return you to an interactive prompt. Once you've executed your further commands, you can then use `fg` to bring the console logging output back to the foreground again.

CREATING A SUPERUSER

You can use our convenience script to create a superuser in the context of the `django` container.

```
s/create-superuser
```

The script will prompt you for required user attributes:

```
Email address: myexampleemail@example.com
Role: 1
First name: Test
Surname: User
Is rcpch audit team member: True
Password:
Password (again):
Superuser created successfully.
```

Notes on superuser creation:

- `Role` - this is an enum which comes from `epilepsy12.constants.user_types.ROLES` and is an integer between 1 and 5
- `Is rcpch audit team member` - this is a boolean value, either `True` or `False`
- `Password` - for superusers in local development it is possible to bypass the minimum password strength requirements, but this is not possible in production environments.

EXECUTING COMMANDS IN THE CONTEXT OF THE `DJANGO` CONTAINER

You can run arbitrary commands in the context of any of the containers using Docker Compose.

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

```
docker compose exec django <command>
```

For example, to send a test email

```
sudo docker compose exec django python manage.py sendtestemail myexampleemail@example.com
```

RUNNING THE TEST SUITE

```
s/test
```


This will execute our suite of Pytest tests inside the `django` container, and the output should be displayed in the console for you.

SHUTTING DOWN THE DOCKER COMPOSE ENVIRONMENT

`^ Ctrl` + `C` will shut down the containers but will leave them built. This means you can restart them rapidly with `s/up`.

To shut down all containers use

```
s/down
```

To shut down **and destroy the containers and the images** they are built from, use

```
s/down-rm-containers-images
```

To go even further and **delete all the data of the application**, including the database, use

```
s/DELETE-LOCAL-DATA
```

For obvious reasons this is something that should **ONLY** be done in local development environments and never on Live! It is named with a different pattern to all the other scripts in order to prevent it accidentally being run.

RESTARTING AND REBUILDING THE CONTAINERS

For convenience and speed we have created some scripts to restart and rebuild the containers.

```
s/restart
```

is the equivalent of running `s/down` followed by `s/up`.

```
s/rebuild
```

is the equivalent of running `s/down-rm-containers-images` followed by `s/up`.

SEEDING WITH DUMMY DATA

For testing of the UI it is often useful to have some dummy data in the database. We have a script to seed the database with some dummy data.

```
s/seed
```

See the Seeding the Database section for more details on the usage of this script, for example setting a non-default Cohort Number.

Tips and Tricks, Gotchas and Caveats

RECONNECTING TO CONTAINERS WHICH ARE ALREADY RUNNING

If you run `docker ps` and it lists the full suite of running E12 containers, and you just want to reconnect to the scrolling log output in the current terminal window, you can just type `s/up` which will not restart anything if they are already running, it will just reconnect to the outputs of those containers so you can see the logs.

EVEN WITH DOCKER, THERE ARE A FEW EXTERNAL LOCAL DEPENDENCIES

Although the Docker Compose setup is very convenient, and it installs all the runtime development dependencies inside the `django` container, one thing it can't do is install any local Python packages which are required for code 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 using `pyenv`. The versions of these dependencies are much less likely to conflict across projects, so you could probably get away with installing them in your system Python if you wished.

DOCKER COMPOSE AND VIRTUAL PRIVATE NETWORKS

If you experience persistent problems with Docker's internal connectivity, make sure you are not using a system-wide Virtual Private Network, or some other tool which might block Docker's internal network traffic. We experienced this problem using **Mullvad** VPN on Linux Mint, on one of our developer team's machines. Docker compose ran fine, and each container appeared

to work independently, but each container was unable to 'see' the others, resulting in crashes and errors. The solution was to disable the VPN.

DOCKER BUILD CACHE ERRORS ON VPS DEPLOYMENTS

On some occasions we have encountered errors when trying to run `s/up` on a VPS, where the Docker build cache is corrupted. This can be resolved by running `docker builder prune` and then `s/up` again. This clears the Docker build cache and forces it to rebuild the images from scratch. Importantly for Live, it does not affect Docker Volumes, so the database and other data is not lost.

5.3.2 Windows Docker Setup

You should have already downloaded Docker and cloned the repository.

Enabling the WSL Terminal within VS Code

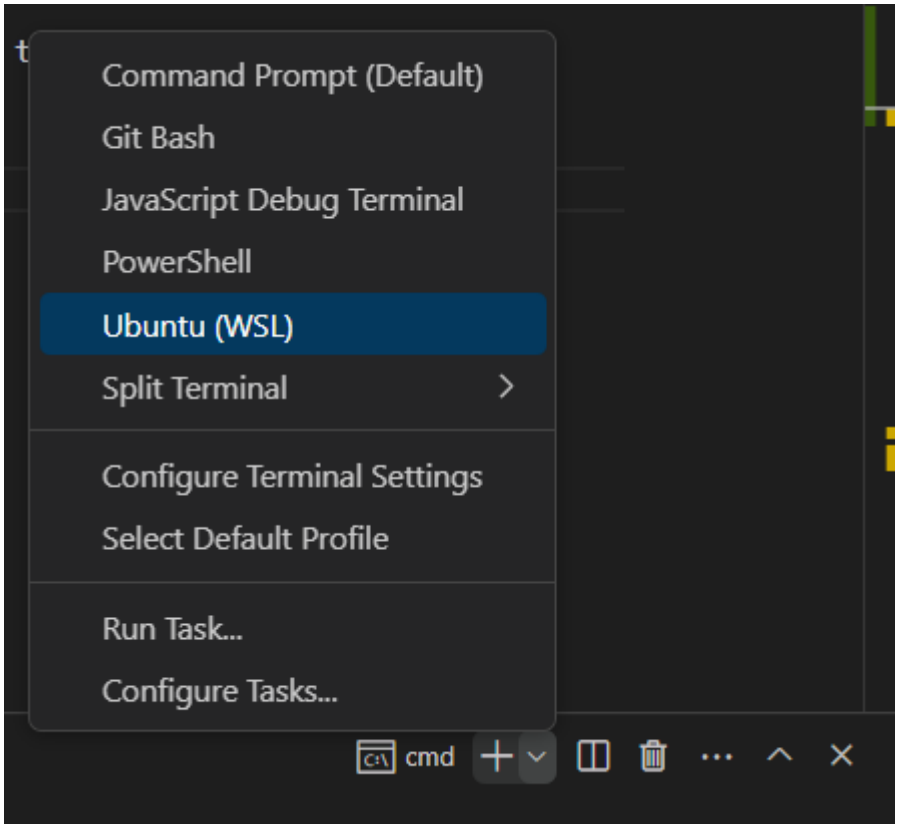
There are scripts in the `s/` which streamline the setup for the development process. Unfortunately, Windows does not natively support running these through the Command Prompt. Instead, we must first install the **Windows Subsystem for Linux (WSL)** to run the scripts.

VS Code has a helpful extension to do just this!

Follow this guide (Windows Subsystem for Linux VSCode Extension) to enable usage of a Ubuntu (WSL) terminal within VS Code.

Running Scripts

Open a new WSL terminal by selecting it:



If you haven't already, `cd` into the root folder

```
cd rcpch-audit-engine/
```

Finally, you should be able to run the setup script by typing:

```
sh s/up
```


Setup errors

Sometimes, the easiest fix for many headaches, relating to installation and setup, is to simply restart your computer and try again!

WSL <> Powershell <> Windows Caddy Certificate

Though WSL is required for the Docker setup, a few additional steps are needed related to Caddy Certificates.

First, ensure all the Docker containers are running by running `sh s/up` inside a WSL2 terminal, as described in the previous step.

To get the Caddy Certificate, open a Powershell terminal in the same directory. Powershell is required because the steps to install the Certificate are OS-specific, and we need the Windows installation.

Verify the containers are running and visible to PS. Open a PS terminal, **as an administrator**, navigate to the project folder and type:

Powershell terminal

```
docker compose ps
```

Which should result in something like:

Powershell terminal

```
NAME                              IMAGE                             COMMAND                  SERVICE             CREATED             STATUS
PORTS
rcpch-audit-engine-caddy-1        caddy                             "caddy run --config …"   caddy               27 minutes ago      Up 19 minutes
0.0.0.0:80->80/tcp, 0.0.0.0:443->443/tcp, 443/udp, 2019/tcp
rcpch-audit-engine-django-1       rcpch-audit-engine-django         "sh -c 'python manag…"   django              27 minutes ago      Up 19 minutes
rcpch-audit-engine-mkdocs-1       rcpch-audit-engine-mkdocs         "sh -c 'mkdocs build…"   mkdocs              27 minutes ago      Up 19 minutes
0.0.0.0:8001->8001/tcp
rcpch-audit-engine-postgis-1      postgis/postgis:15-3.3            "docker-entrypoint.s…"   postgis             27 minutes ago      Up 19 minutes
5432/tcp
```

Next, type:

Powershell terminal

```
docker compose cp caddy:/data/caddy/pki/authorities/local/root.crt %TEMP%/root.crt
```

Which will copy the `root.crt` from inside the `caddy` container into `%TEMP%`.

Finally, install the certificate by typing:

Powershell terminal

```
certutil -addstore -f "ROOT" %TEMP%/root.crt
```

Close and re-open Chrome, and everything should be working!

5.4 Manual 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.

5.4.1 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 `envs/env-template` 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
```

5.4.2 Install the correct Python version

If you don't have Python installed already, you will need it. To avoid specifying a specific Python version for the project here in the documentation, please check the `Dockerfile` in the project root for the version of Python that is currently being used.

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

```
pyenv install <PYTHON_VERSION>
```

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 <PYTHON_VERSION> 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
```

5.4.3 Set and initialize the environment variables

You will need to set the environment variables for your local development, using the `envs/env-template` file as a starting point. **This file should not be committed to the repository**. You can use real values for the environment variables in this file.

```
source envs/.env
```

Danger

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

5.4.4 Prepare the database for use

```
python manage.py migrate
```

5.4.5 Create superuser to enable logging into admin section

```
python manage.py createsuperuser
```

Then follow the command line prompts to create the first user. Createsuperuser is a Django base feature but there are some custom fields which are mandatory. These include:

- `role`: The options are: `1 - Clinical Lead, 2 - Clinician, 3 - E12 Site Administrator, 4 - RCPCH Audit Team`. If the integer selected in 1-3 (ie a role within the E12 site) KCH is automatically allocated. If it is an RCPCH user, `is_rcpch_staff` is automatically set to true, as is `is_rcpch_audit_team_member`
- `is_rcpch_audit_team_member`: `True|False`.

Further users can subsequently be created in the Admin UI

5.4.6 Running the server

Navigate to the epilepsy12 outer folder and run the server:

```
python manage.py runserver
```

5.4.7 Seeding the Database

Migrations will seed the database with the following:

- Organisations
- Trusts
- ICBs (including boundaries)
- NHS England regions (including boundaries)
- Countries (including boundaries)
- KPI aggregations (the tables required to store the results generated by counting up KPIs for individual children at different levels of abstraction eg trust/icb etc)
- Epilepsy causes, syndromes and comorbidities: these are all pulled from SNOMED and persisted in their own tables
- Semiology keywords. These are ILAE 2017 keywords to standardise the description of a seizure. It is not an exhaustive list and in time it is hoped through this project that new meaningful words will be identified and add to this lexicon.

By running the migrations there database will therefore be ready to accept new users and children.

In development, it is often necessary to have some seeded children across multiple organisations to test functionality, so in addition there are 2 further seed commands:

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

This accepts the `-c` or `cases` attribute followed by the number of children requested. It defaults to 200. Children are given random dates of birth, NHS numbers and postcodes which are all valid.

Once the children have been seeded, they can be scored at random:

```
python manage.py seed --mode seed_registrations
```

This accepts the `-ct` or `--cohort` attribute followed by the cohort number requested. Note that it is not possible to seed with patients below cohort 4. It defaults to the current actively recruiting cohort.

Both functions accept the `--verbose` flag also to have a more granular output to the console of the seeding process for debugging purposes.

There are some other functions here also, but these are likely soon to be deprecated:

- add_permissions_to_existing_groups
- upload_old_patient_data
- async_upload_old_patient_data
- upload_user_data
- async_upload_user_data

This will seed with the defaults documented above.

5.4.8 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.

5.5 Testing

5.5.1 Running Tests

Tests for the Epilepsy12-specific parts of the platform are organised in an `epilepsy12/tests/` folder inside the epilepsy12 app. We have opted to use Pytest, which is well-regarded in the Django community.

> ⚠️ **Active Docker Container**
>
> Please ensure that your Docker container is still built and active. The previous command in the 'Docker setup' page was to illustrate how to close the Docker container. To reopen it, run:
>
> ```
> s/up
> ```

Running `pytest`

When running tests, it is important to understand that they will only run **inside** the Docker container (assuming you have used the Docker development setup). Therefore, how you run the tests depends on whether you are using Docker Desktop (either through the native application or VSCode extension where you can attach a shell terminal to the Docker environment) or Docker Compose. The following examples assume you are at the root of the project.

Using Docker Desktop **Using s/docker scripts**

Using the integrated terminal in Docker Desktop:

```
pytest
```

Run the following command in your normal system terminal:

```
s/test
```


5.5.2 Customisation and Useful Flags

`pytest.ini`

The `pytest.ini` file in the project root defines configurations for our tests.

Take this example:

```
[pytest]

# POINT PYTEST AT PROJECT SETTINGS
DJANGO_SETTINGS_MODULE = rcpch-audit-engine.settings

# SET FILENAME FORMATS OF TESTS
python_files = test_*.py


addopts =
    --reuse-db # RE USE TEST DB AS DEFAULT
    -k "not examples" # AVOID TESTS MARKED AS EXAMPLES

markers =
    examples: mark test as workshop-type / example test
    seed: mark test as 'meta test', just used for seeding
```

Configuration Option	Description
`addopts`	Adds arguments to the `pytest` command by default. E.g. running `pytest` actually results in: `pytest --reuse-db -k "not examples"`
`markers`	Define additional markers to selectively run tests.

Markers

Pytest allows the use of 'mark'ing tests. Custom marks can be added in `pytest.ini`.

Marks are applied by using the Pytest decorator over test functions, which can be chained e.g.:

```
@pytest.mark.xfail
@pytest.mark.parametrize(
    "DATE_OF_BIRTH,REGISTRATION_DATE,TIMELY_MRI,EXPECTED_SCORE",
    [
        (date(2022, 1, 1),date(2024, 1, 1),True,KPI_SCORE["PASS"],),
        (date(2022, 1, 1),date(2024, 1, 1),False,KPI_SCORE["FAIL"],),
        (date(2022, 1, 1),date(2024, 1, 2),True,KPI_SCORE["NOT_APPLICABLE"],),
    ],
)
@pytest.mark.django_db
def test_measure_4_mri_under2yo(
    ...
)
```

Default marks are included with Pytest. Some of these include:

Mark	Description
`@pytest.mark.xfail`	Mark which labels test as expected fail, used for Test-Driven-Development where tests are written before code.
`@pytest.mark.django_db`	Mark which enables test-db access.
`@pytest.mark.parametrize`	Mark which parametrizes test functions.

Selecting tests through name

The `-k` flag will select and run tests based on the provided substring.

For example, the following commands will collect and run tests starting with `test_measure_1`:

Using Docker Desktop **Using Docker Compose**

```
pytest -k "test_measure_1"

docker compose -f docker-compose.yml exec web pytest -k "test_measure_1"
```

Skipping or selecting tests

Some of the more time-consuming tests can be skipped for convenience using the 'mark' feature in Pytest. For example, the following will **skip** tests marked 'examples'.

Using Docker Desktop **Using Docker Compose**

```
pytest -k "not examples"

docker compose -f docker-compose.yml exec web pytest -k "not examples"
```

Verbosity

You can increase the verbosity of the Pytest output using the `-v` flag. The more `-v` flags you add, the more verbose the output, to a limit of -vv. The following will run the tests with the maximum verbosity.

Using Docker Desktop **Using Docker Compose**

```
pytest -vv

docker compose -f docker-compose.yml exec web pytest -vv
```

Running a single test

Pytest allows a specific test to be run by specifying the path to the test file. This can be useful when debugging a particular test. For example, the following will run only the tests in the file at the specified path.

Using Docker Desktop **Using Docker Compose**

```
pytest epilepsy12/tests/model_tests/test_antiepilepsy_medicine.py

docker compose -f docker-compose.yml exec web pytest epilepsy12/tests/model_tests/test_antiepilepsy_medicine.py
```

To run **one specific test** within this file, use the `PATH_TO_TEST_FILE::TESTNAME` notation:

Using Docker Desktop **Using Docker Compose**

```
pytest epilepsy12/tests/model_tests/test_antiepilepsy_medicine.py::test_length_of_treatment

docker compose -f docker-compose.yml exec web pytest epilepsy12/tests/model_tests/test_antiepilepsy_medicine.py::test_length_of_treatment
```

Capturing `stdout` and `stderr`

If you have used `print()` statements in your code, you may want to capture the output of these statements in your tests. By default, Pytest captures the stdout from tests and displays them depending on certain conditions.

You can specify how stdout is displayed using the `-s` and/or `-rP` flags.

The `-rP` flag defines how Pytest shows the "short test summary info" content. It combines the `-r` option, which shows "failures and errors" by default, with `P`, which adds the captured output of passed tests.

Using Docker Desktop **Using Docker Compose**

```
pytest -rP

docker compose -f docker-compose.yml exec web pytest -rP
```


The `-s` flag will tell Pytest to not capture the stdout and instead prints it straight to the console:

Using Docker Desktop **Using Docker Compose**

```
pytest -s

docker compose -f docker-compose.yml exec web pytest -s
```

> **`pytest -h` for help**

Further details on the options available can be found by running `pytest -h` and visiting the Pytest documentation.

5.5.3 Test Structure

Global Pytest configuration settings are set within `pytest.ini`, in the top-level directory of the Django project.

Tests related to Epilepsy12 are found in the `epilepsy12/tests` directory.

This is an overview of their contents.

Factories, fixtures, meta tests

Module	Description
`_meta_tests`	Folder containing tests related to tests concerning the actual test environment setup, such as test-db seeding.
`factories`	Module containing all FactoryBoy factories.
`conftest.py`	Defines global fixtures.

Test directories

We aim for 100% test coverage. To this end, our test directory name-spacing mirrors that of the E12 App.

Directory	Description
`common_view_functions_tests`	Folder containing tests related to the `common_view_functions`.
`general_functions_tests`	Folder containing tests related to the `general_functions_tests`.
`model_tests`	Folder containing tests related to the `model_tests`.
`view_tests`	Folder containing tests related to the `view_tests`.

5.5.4 Test Database

The test database used by Pytest will be automatically set up the first time `pytest` is run. The test database mirrors E12's database, applying the same migrations and seeding functions, including:

- Seeding Cases
- Seeding Groups & Permissions

This initial setup is scoped to once per session and persists between subsequent test runs. In practice, this means the first time you run `pytest`, it may take a few minutes to set up, but subsequent runs should be much quicker as the pre-seeded database will be used.

> **Conservative Database Access**
>
> By default, Pytest does not allow database access in its tests.
>
> If you require a test to access the database, mark it with the `@pytest.mark.django_db` decorator.
>
> Alternatively, you can feed it in as a fixture. However, we only use marks for consistency.

5.5.5 Factories and Fixtures

Pytest and FactoryBoy factories and fixtures enable DRY code, allowing similar object instances to be defined once, and shared many times across the test directory, enabling faster and more consistent test-driven development.

For example, many of the tests rely on the creation of a fully-completed audit, which is comprised of the following:

1. `Case` model with linked `Organisation`(s), through the `Site` conduit model
2. Linked `Registration` model

a. Linked `AuditProgress` model

b. Linked `KPI` model

c. Linked `FirstPaediatricAssessment` model

d. Linked `EpilepsyContext` model

e. Linked `MultiaxialDiagnosis` model

f. Linked `Assessment` model

g. Linked `Investigations` model

h. Linked `Management` model

Following the best-practice principle of DRY code, we use FactoryBoy factories to define the creation of Epilepsy12 `Case` instances, with default values set once, minimising repeated code. Whenever an object is created using the 'top-level' `e12_case_factory`, all the linked dependency models have generated automatically, with default values that can be overridden.

`conftest.py`

The `conftest.py` file registers fixtures globally to be used in any test without import. Available factories are imported and registered like so:

```
from epilepsy12.tests.factories import (
    E12CaseFactory,
    ... # other factories
)

register(E12CaseFactory)  # => e12_case_factory
```

When factories are registered this way, their default fixture name becomes the lowercase, under-score version of the class name. In this case, `E12CaseFactory`'s fixture becomes `e12_case_factory`, which can be used like so:

```
@pytest.mark.django_db
def test_example(e12_case_factory):

    fully_completed_case = e12_case_factory()
```

> **Conservative Database Access**
>
> By default, Pytest does not allow database access in its tests.
>
> If you require a test to access the database, mark it with the `@pytest.mark.django_db` decorator.
>
> Alternatively, you can feed it in as a fixture. However, we only use marks for consistency.

Using E12Factories

As factories have been registered within `conftest.py`, whenever a factory is required in a test, pass in the lowercase, under-scored fixture name into the test arguments:


```
@pytest.mark.django_db
def test_example(
    e12_case_factory
):
    ...
```

Each E12 model has an associate factory, whose fixtures are namespaced using the `e12_LOWERCASE_UNDERSCORE_MODELNAME_factory` e.g. pattern:

```
register(E12AntiEpilepsyMedicineFactory)  # => e12_anti_epilepsy_medicine_factory
register(E12AssessmentFactory)  # => e12_assessment_factory
register(E12CaseFactory)  # => e12_case_factory
register(E12ComorbidityFactory)  # => e12_comborbidity_factory
register(E12EpilepsyContextFactory)  # => e12_epilepsy_context
register(E12EpisodeFactory)  # => e12_episode_factory
register(E12FirstPaediatricAssessmentFactory)  # => e12_first_paediatric_assessment_factory
register(E12ManagementFactory)  # => e12_management_factory
register(E12MultiaxialDiagnosisFactory)  # => e12_multiaxial_diagnosis_factory
register(E12RegistrationFactory)  # => e12_registration_factory
register(E12SiteFactory)  # => e12_site_factory
register(E12SyndromeFactory) # => e12_syndrome_factory
register(E12UserFactory)  # => e12_user_factory
```

USAGE

For most test cases, which require multiple different linked models (e.g. a `Registration` attached to a `MultiaxialDiagnosis`), you should instantiate starting from the `e12_case_factory`:

```
@pytest.mark.django_db
def test_example(
    e12_case_factory
):
    case = e12_case_factory()
```

This will create and save a fully-registered and audit-complete `Case`, accessible using the `case` variable.

ACCESSING VALUES AND OVERRIDING DEFAULTS

Default attributes of the `Case` model can be overridden:

```
@pytest.mark.django_db
def test_example(
    e12_case_factory
):
    case = e12_case_factory(
        first_name = "Bob",
        surname = "Dylan",
    )
```

Dependency factory attributes can be directly overridden, up to 2 dependencies down (the factories directly related to a `Registration`), using the dunder-format:

```
@pytest.mark.django_db
def test_example(
    e12_case_factory
):
    case = e12_case_factory(

        # Case.first_name
        first_name = "Bob",

        # Case.surname
        surname = "Dylan",

        # Registration.registration_date
        registration__registration_date=date(2023,1,1),

        # Assessment.childrens_epilepsy_surgical_service_referral_criteria_met WHERE Assessment.registration is linked to this instance
        registration__assessment__childrens_epilepsy_surgical_service_referral_criteria_met=True,

        # Investigations.mri_brain_requested_date WHERE Investigations.registration is linked to this instance
        registration__investigations__mri_brain_requested_date=date(2023,4,1)
    )
```

Values can be directly accessed up to 2 dependencies down:

```
print(case.first_name) # => Bob
print(case.registration) # => Epilepsy12 registration for Bob Dylan on 2023-01-01
print(case.registration.multiaxialdiagnosis) # => Multaxial diagnosis for Bob Dylan
print(case.registration.multiaxialdiagnosis.syndrome_present) # => True
```

Accessing `Multiaxial_Diagnosis` attribute

NOTE: to access `Multiaxial_Diagnosis` in this way, use the lowercased, non-underscored name: `multiaxialdiagnosis`

These all return Django model objects, which can be used in the classic Django way if further customisation is required for tests:

epilepsy12/tests/common_view_functions_tests/calculate_kpi_tests/test_measure_5.py

```
from epilepsy12.models import (
    Registration,
    Syndrome,
)

@pytest.mark.django_db
def test_measure_4_mri_syndromes_ineligible(
    e12_case_factory,
    ...
):

    ...

    case = e12_case_factory()

    # get registration for the saved case model
    registration = Registration.objects.get(case=case)

    # get syndrome for registration
    current_syndromes = Syndrome.objects.get(
        multiaxial_diagnosis=registration.multiaxialdiagnosis
    )

    ...
```

FLAGS

Specific factories contain flags that override multiple values if set `True`. For example, the `E12_Assessment_Factory` assigns `consultant_paediatrician_referral_made=True` by default, with `consultant_paediatrician_referral_date` and `consultant_paediatrician_input_date`.

The `Assessment.no_referral_consultant_paediatrician=True` flag can be used to override this:

```
@pytest.mark.django_db
def test_testing(
    e12_case_factory
):

    case = e12_case_factory(
        registration__assessment__no_referral_consultant_paediatrician=True
    )

    print(case.registration.assessment.consultant_paediatrician_referral_made) # => False
    print(case.registration.assessment.consultant_paediatrician_referral_date) # => None
    print(case.registration.assessment.consultant_paediatrician_input_date) # => None
```


5.5.6 Faking Values

There are multiple reasons we may wish to patch values in our tests, including:

- Simulating external service responses
- Controlling return values
- Error handling
- Speeding up tests

For example, `test_registration_days_remaining_before_submission` relies on calculating the difference in days between `datetime.now().date()` and `Registration.audit_submission_date`.

Of course, `datetime.now().date()` will always change, leading to the tests breaking at some eventual future time point.

Instead, we can temporarily patch the return value of `datetime.now().date()`, **only within the scope of this test**, with the `@patch.object()` decorator, using the following syntax:

```
from unittest.mock import patch

@patch.object(CLASS_NAME_TO_PATCH, 'CLASS_METHOD_NAME_TO_PATCH', return_value=VALUE_TO_RETURN)
```

In action, this looks like this:

test_registration.py

```
from unittest.mock import patch

...

@patch.object(Registration, 'get_current_date', return_value=date(2022, 11, 30))
@pytest.mark.django_db
def test_registration_days_remaining_before_submission(
    mocked_get_current_date,
    example_fresh_registration,
):

    ...
```

We assign the return value of `Registration.get_current_date` to always be 2022-11-30.

Using the `@patch.object` decorator also passes in the patched object into the test function as the FIRST parameter, which can be called anything - in this case, we call it `mocked_get_current_date`. Other fixtures required for the test are passed after.

If you use multiple `@patch.object` decorators, patched objects are passed in the same order decorators are evaluated:

test_registration.py

```
from unittest.mock import patch

@patch.object(OBJECT1, 'method_to_patch', return_value=1)
@patch.object(OBJECT2, 'method_to_patch', return_value=3)
@patch.object(OBJECT3, 'method_to_patch', return_value=3)
def test_my_test(
    patched_object_3,
    patched_object_2,
    patched_object_1,
):
    ...
```

5.5.7 Writing Tests

Test Structure

Test files are stored in appropriate directories, mirroring the E12 App. Please see Test Structure for details.

The basic structure for test files is as follows:

```
"""
DOCSTRING GIVING AN OVERVIEW OF THE CONTAINED TESTS
"""

# Standard imports
import pytest

# Third party imports

# RCPCH imports


def test_function_1():
    """
    DOCSTRING DESCRIBING TEST SPECIFICS
    """

    # Test Arrange steps
    ...

    # Test Act steps
    ...

    # Test Assert steps
    assert True == False, "REASON FOR FAILURE"
```

Real Example

Using a real example:

epilepsy12/tests/model_tests/test_case.py

```
"""
Tests the Case model
"""

# Standard imports
import pytest
from datetime import date

# Third party imports

# RCPCH imports

@pytest.mark.django_db
def test_case_age_calculation(e12_case_factory): # (4)
    # Test that the age function works as expected # (3)

    e12Case = e12_case_factory() # (1)

    fixed_testing_date = date(2023, 6, 17)
    e12Case.date_of_birth = date(2018, 5, 11)

    assert e12Case.age(fixed_testing_date) == "5 years, 1 month", "Incorrect stringified age" # (2)
```

1. Arrange & Act Steps: Create necessary Case dependency for test, alongside setting date values to test.
2. Assert Step: asserts the stringified age is correct, with assertion error message.
3. Specific explanation of test.
4. ⚠ Test names MUST start with `test_` for them to be detected by Pytest.

5.5.8 Best Practices

Descriptive Test Function Names

Tests should have clear and descriptive names conveying purpose. This improves the readability of the test suite.

Test independence and isolation

Tests should be independent, never relying on the outcomes of other tests.

Unit vs Integration Tests

Unit tests focus on testing individual components in isolation, such as Model Tests.

Integration tests focus on interactions between multiple components, such as the `calculate_kpi` tests.

New code needs new tests

Ideally, following Test-Driven-Development practice, tests should be written before the code is written.

All new code requires tests to be written to be confident of functionality, and prevent future regressions.

New PRs must pass all tests.

5.5.9 Coverage

Using the `coverage` tool, we can get some code analysis, including the total test coverage. For example, the following will run pytest through `coverage`. Then type `coverage report` to see the coverage report.

Using Docker Desktop **Using Docker Compose**

```
coverage run -m pytest
coverage report

docker compose -f docker-compose.yml exec web coverage run -m pytest
docker compose -f docker-compose.yml exec web coverage report
```


5.6 Code Style

5.6.1 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

5.6.2 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`

5.6.3 Formatter

5.6.4 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.

5.7 RCPCH Branding

5.7.1 Logos and Branding

The RCPCH Design Team have given us help and advice in ensuring the RCPCH Incubator builds products which adhere to RCPCH Brand Guidance.

In addition they have created custom logos for our projects and even our teams. These logo files are present in the `docs/_assets/_images` folder of this repository.

5.7.2 RCPCH Brand Guidance

The RCPCH full brand guidance can be found here (private repository, access for RCPCH team only)

5.8 Application Structure

5.8.1 Overall structure

The RCPCH Audit Engine is a generic framework for national clinical audits. Its first deployment is as a new platform for the RCPCH's established Epilepsy12 audit, but it is designed to be reusable for other audits in the future.

National clinical audits 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 at national level.

Project Design

The RCPCH development team used Django, a Python-based web framework which is mature, accessible and well-documented. It is founded on the concept of a Project which can have many Applications within it. This meant that we could have an `rcpch-audit-engine` project, within which multiple audit applications might sit, sharing resources, for example relating to authorisation 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 Project, currently `epilepsy12` is the only Application.

5.8.2 Design 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.

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.

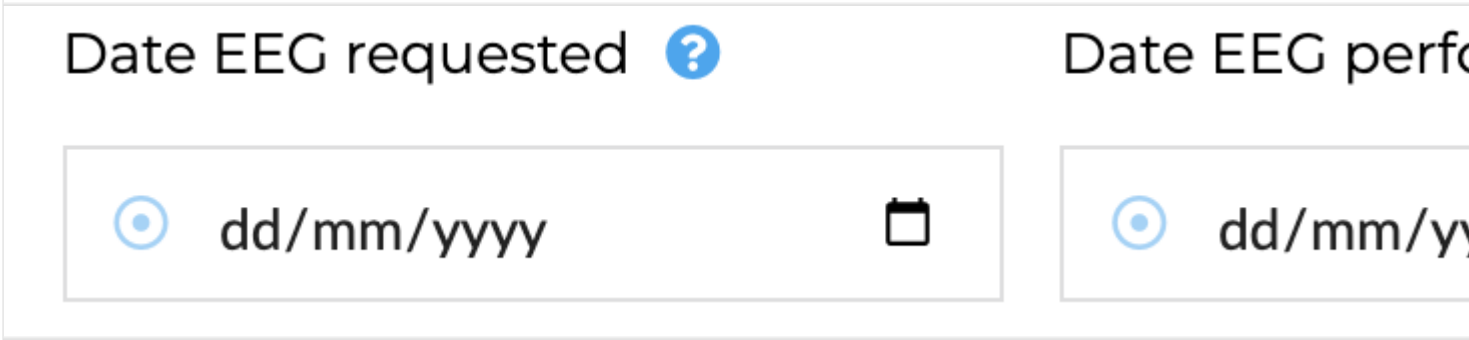
- `base.html` : This forms the base of all content and is used as a parent template for most content. It includes the `<head>` tag and any javascript/jquery on which semantic ui is dependent.
- `audit_section.html` contains the structure of each form and has blocks within it for the text of the headers 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`

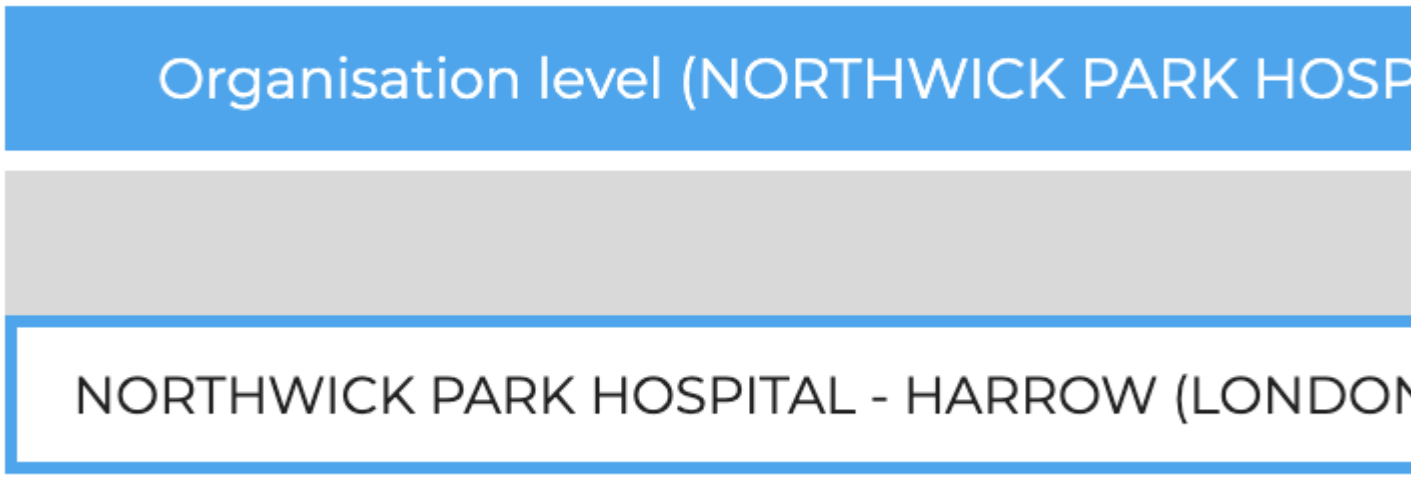
This is a subfolder of `partials` and needs extra explanation, as this contains the customized widgets that make up the RCPCH audit forms. The widgets are named as follows:

Page Elements

date_field.html

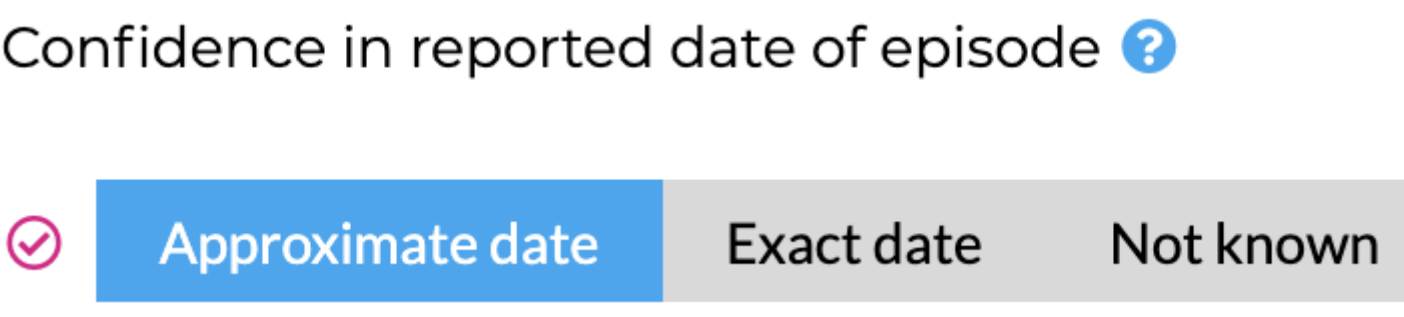


rcpch_multiple_toggle.html



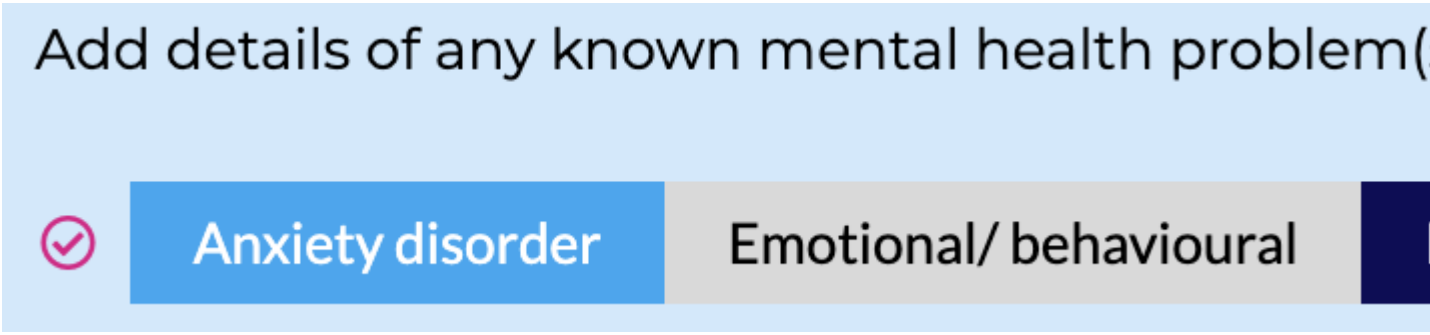
toggle_button.html

single_choice_multiple_toggle_button.html

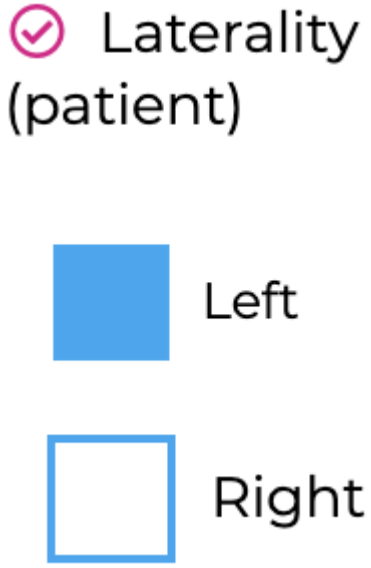


multiple_choice_multiple_toggle_button.html

Page Elements



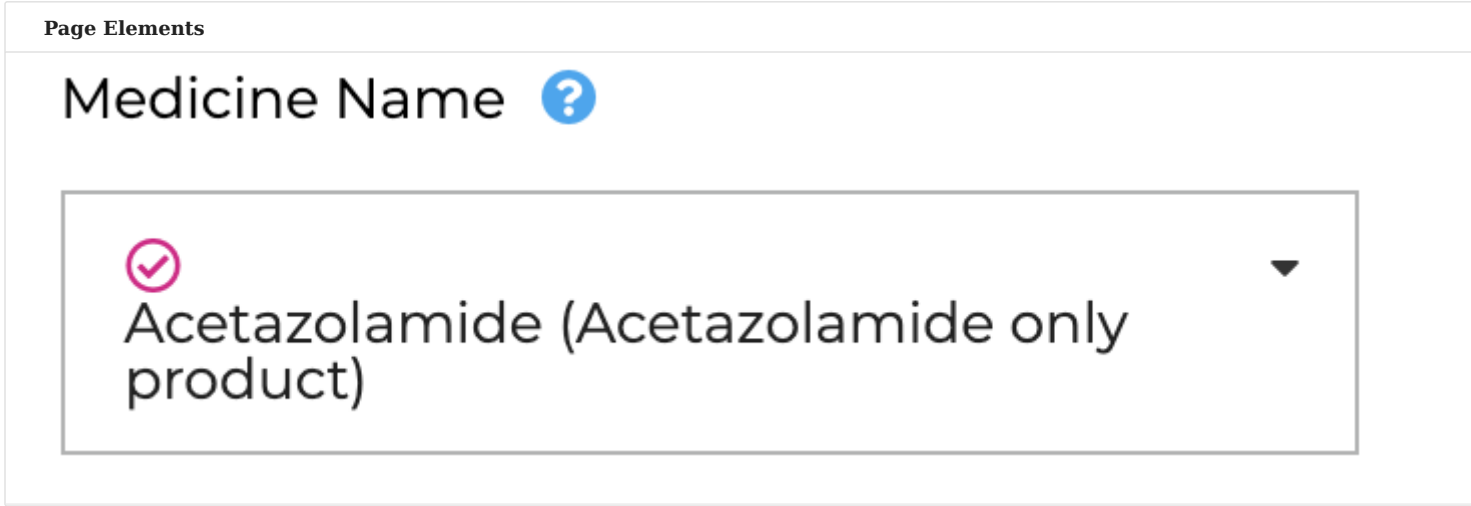
checkbox_group.html



rcpch_organisations_select.html

select_model.html

Page Elements



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

tooltip_id: **multiple_choice_multiple_toggle** id of the element for tooltip text

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. registration_dates.html: This partial is found in `templates/epilepsy12/partials/registration/registration_dates` 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 is sent in the header from the view to the template, and is triggered in the template. 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
```

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.

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.

5.8.3 Database

Frameworks

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

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:

BASE TABLES

- **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.
- **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.
- **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.
- **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.
- **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.
- **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. Information is stored here about start date, whether the medicine is for rescue, and whether aspects relating to side-effects and so on have been discussed. It has a one to many relationship with MedicineEntity.

REPORTING TABLES

Tables also track the progress of each child through the audit, as well how they are scoring with regard to their key performance indicators (KPIs). These KPIs are aggregated periodically to feed reports on KPIs at different levels of abstractions (organisational, trust-level or health board, integrated care board, NHS England region and country level)

- **AuditProgress**: Has a one to one relationship with Registration. Stores information on how many fields in each form have been completed.
- **KPI**: scores each individual child against the national KPI standards. It stores information on whether a given measure has been passed, failed, has yet to be completed, or whether the child in not eligible to be scored.
- **KPIAggregation**: This base model stores results of aggregations of each measure. The base model is subclassed for models representing each geographical level of abstraction. Aggregations are run at scheduled intervals asynchronously and pulled into the dashboards.
- **VisitActivity**: Stores user access/visit activity, including number of login attempts and ISP address as well as timestamp

LINK TABLES

There are some many to many relationships. Django normally handles this for you, but the development team chose to implement the link tables in these cases separately to be able to store information about the relationship between the tables.

- **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 Organisation trust model (since one Organisation can have multiple Cases and one Case can have multiple Organisations). 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 organisation 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).
- **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. The look up table for Comorbidity is ComorbidityEntiy, which is seeded from SNOMED. This allows a many to many relationship between MultiaxialDiagnosis and CormorbidityEntity
- **AntiEpilepsyMedicine**: is a link table between Management and MedicineEntity

LOOKUP TABLES

These classes are used as look up tables throughout the Epilepsy12 application. They are seeded in the first migrations, either pulling content from the the `constants` folder, or from SNOMED CT.

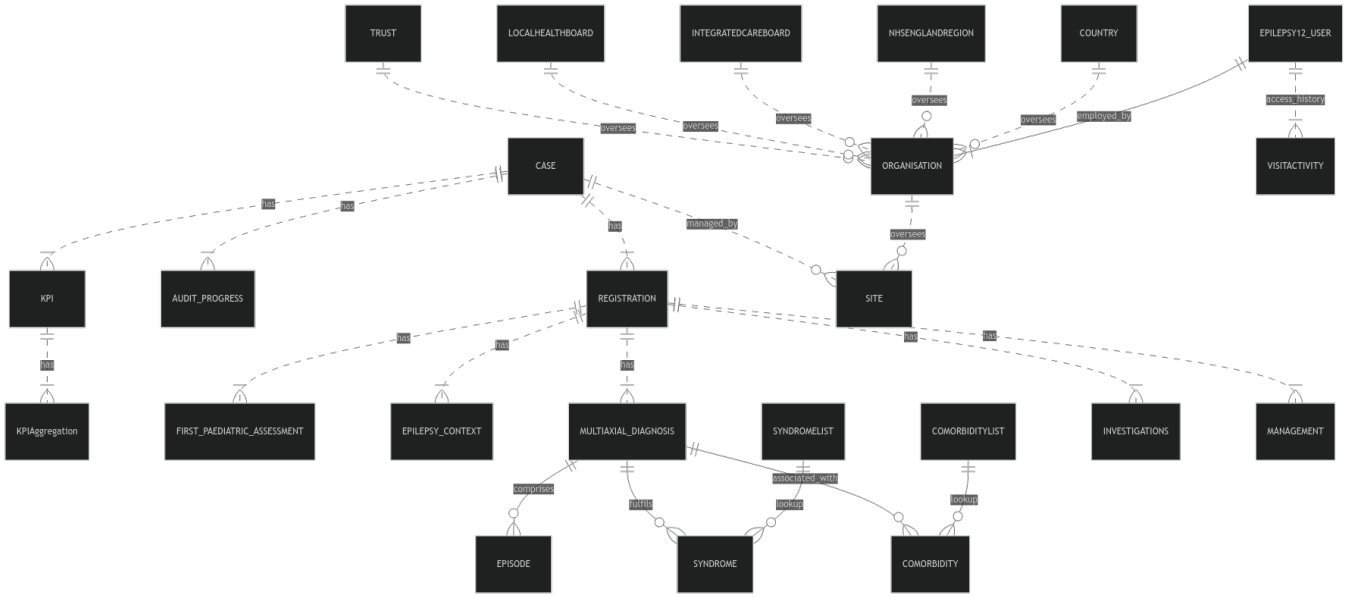
- **Organisation**: This model stores information about each Organisation in England, Scotland and Wales. It is used as a lookup for clinicians as well as children in Epilepsy12. It has a many to many relationship with Case and a many to one relationship with Epilepsy12User. It is seeded from the `constants` 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.
- **EpilepsyCause**: Seeded from `constants`, provides a look up for MultiaxialDiagnosis.
- **MedicineList**: Seeded from SNOMED, provides lookup for the AntiepilepsyMedicine model.
- **SyndromeList**: Seeded from SNOMED, provides lookup for the MultiaxialDiagnosis model.
- **IntegratedCareBoard**: Seeded from `constants` provides a list of Integrated Care Boards and identifiers
- **OpenUKNetwork**: Seeded from `constants` provides a list of OPENUK Networks and identifiers
- **LocalHealthBoard**: Seeded from `constants` provides a list of Local Health Boards in Wales and identifiers
- **NHSEnglandRegion**: Seeded from `constants` provides a list of NHS England regions and identifiers
- **Country**: Seeded from `constants` provides a list of country identifiers

Boundary files and geography extension pack

We have included the Django GIS extension allowing geographic data to be stored. This allowed for `.shp` files for the different regions to be stored and mapping therefore to be possible. The `.shp` files are stored in the following models:

- **IntegratedCareBoard**
- **LocalHealthBoard**
- **NHSEnglandRegion**
- **Country**

In future it is planned to use anonymised, aggregated patient postcodes to calculate distance to epilepsy treatment centres and correlate this against Key Performance Indicators.



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. The initial migrations contain seed functions to seed the database on creation with data required for the lookup tables listed above.

5.8.4 Dropdown Lists

Lists

Lists that feed either the toggle buttons or the select dropdowns are either found in the `constants` folder, or are seeded to the database in the `migrations`.

Organisation lists are seeded from `constants` as are the levels of abstraction associated with them (trust, ICB etc.)

The **Epilepsy causes** and **comorbidities**, by contrast are a SNOMED refset seeded from the RCPCH SNOMED server in migration 0006 and 0009 into EpilepsyCause and ComorbidityList respectively.

Syndromes are from lists in the `constants` folder.

Epilepsy medicines (both rescue and not) are found in the `constants` folder but look up the full SNOMED concept from the RCPCH SNOMED server before saving in the Medicine table.

Semiology Keywords, used to categorize words in the free text descriptions of a seizure event, are taken from a list in the `constants` folder.

The tables that supply the dropdowns are seeded in the migrations which is a recommended way in the Django documentation to add data, known as data migrations.

Since go-live, E12 have wished on the basis of user feedback, from time to time to add new items to these lists. The process for adding new items should be:

EPILEPSYCAUSE

1. Epilepsy12 team to supply the SNOMED CT ID (SCTID) of the concept
2. The development team add the SCTID to the list `extra_concept_ids` in migration 0006 for future seeding from scratch, mostly for development reasons
3. In the python shell to run the seed function:

```
docker compose exec django python manage.py seed --mode=add_new_epilepsy_causes -sctids 764946008 52767006 ...
```

Note that the function expects a list, even if only one item is supplied.

ORGANISATIONS

Just updating the `RCPCH_ORGANISATIONS` constant will not in itself update the database, but is a necessary step in the process. The workflow needs to be:

Deleting an organisation

1. check there are no children associated with this organisation. If there are, it must not be deleted
2. Delete the organisation in the admin. This will delete any relationships it also has with associated trusts/health boards etc as well KPIAggregation models

Updating an organisation

This can be done reasonably straightforwardly in the admin. Note that the ODS Code is a unique identifier and if the update includes an update to this, you are in effect creating a new organisation, rather than editing an existing one. Better therefore to create a new organisation and delete the old. This becomes more complicated if there are children associated with this organisation.

Adding a new organisation

This can be done in the admin. The ODS Code must be unique. The name and ODS code should ideally be mandatory fields but are not currently prescribed as such in the model. Note that you must also allocate the Trust/Health Board, ICB, NHS England Region, London Borough and Country. Add the same details to the `RCPCH_ORGANISATIONS` constant. This is necessary later when seeding the KPI Aggregation models in the shell:


```
from epilepsy12.common_view_functions import _seed_all_aggregation_models
_seed_all_aggregation_models()
```

5.8.5 Audit Forms

The Epilepsy12 platform has a reactive user interface with RCPCH colours and design elements. Each audit form relates to an aspect of the child's epilepsy journey to be captured in the audit process. The elements are customised and a deliberate decision was to favour toggle buttons (either single or multichoice) that could be selected easily, were unambiguous and provided structured data easy for analysis.

There is an audit form for each model and each follows the same structure:

`audit_section.html` divides the page into a left side-bar which contains a progress wheel reporting how many fields have been completed, and as series of steps which serve as navigation menu to the different forms that must all be completed prior to submission. They are numbered but do not have to be completed in order. A completed form renders as a pink tile in the steps. A selected tile renders as dark blue. The final tile links to a table of Key Performance Indicators for the care of that individual child.

In the main window, a square segment contains the audit form, and comprises a top-attached header segment with the child's identifiers. The main segment contains the `audit_section_form` block, which populates with the different forms from the `templates/epilepsy12/forms` folder. Finally the footer section contains user information guiding.

Steps

The steps element described above is updated every time an item in the audit form is scored.

```
<div
      hx-get='{% url "registration_active" case_id active_template %}'
      hx-trigger='registration_active from:body'
      hx-target='#registration_active'
      hx-swap="innerHTML"
      name="steps"
      class="rcpch_steps_wrapper"
    >
```

The steps are wrapped in this div which defines a custom `htmx-trigger`, named `registration_active`, called from the `body` element. This HTMX action can be called from any other element and triggers a GET request to the `registration_active` endpoint, found in `views.py`.

```
# 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
    site = Site.objects.filter(
        site_is_actively_involved_in_epilepsy_care=True,
        site_is_primary_centre_of_epilepsy_care=True,
        case=registration.case,
    ).get()
    organisation_id = site.organisation.pk

    # 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,
        "organisation_id": organisation_id,
    }

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

This function retrieves user progress from the `AuditProgress` model and passes this to the steps for it to render progress and rerender the `steps.html` partial with the updated data.

5.8.6 Form Scoring

This is the process of tracking user progress through scoring all the elements of the child's journey through the epilepsy12 audit. Progress is tracked in the `AuditProgress` model which has the following fields:

- `registration_complete`
- `registration_total_expected_fields`
- `registration_total_completed_fields`
- `first_paediatric_assessment_complete`
- `first_paediatric_assessment_total_expected_fields`
- `first_paediatric_assessment_total_completed_fields`
- `assessment_complete`
- `assessment_total_expected_fields`
- `assessment_total_completed_fields`
- `epilepsy_context_complete`
- `epilepsy_context_total_expected_fields`
- `epilepsy_context_total_completed_fields`
- `multiaxial_diagnosis_complete`
- `multiaxial_diagnosis_total_expected_fields`
- `multiaxial_diagnosis_total_completed_fields`
- `investigations_complete`
- `investigations_total_expected_fields`
- `investigations_total_completed_fields`
- `management_complete`
- `management_total_expected_fields`
- `management_total_completed_fields`

For each form, a boolean flag tracks if the form is complete, how many fields in the form have been completed so far as an integer, and how many are expected, also as an integer. This is because the denominator is dynamic - the minimum number of fields expected to be scored to complete the form changes based on the user choices. For example, in `MultiaxialDiagnosis`, if the user selects 'yes' to 'Is there an identifiable epilepsy syndrome?', they are invited to add a syndrome and the date of diagnosis: this increases the number of expected fields therefore by 2.

The final step, is to update the `AuditProgress` with these results and call the `calculate_kpis()` function.

`total_expected_fields` vs `total_completed_fields`

The logic calculating user progress can be summarise by these two fields and is found in `common_view_functions/recalculate_form_generate_response.py`.

`TOTAL_COMPLETED_FIELDS`

This is calculated from two functions:

- `completed_fields()`
- `number_of_completed_fields_in_related_models()`

Both functions accept a registration instance, from which all models can be accessed.

This function loops through all fields in a given model instance counting up all the fields that are not None (since all fields are None until scored). Several fields in each model have to be excluded, since they are not storing information about the audit. These include fields such as the primary key. This stripping of fields occurs in `fields_to_avoid()`.

Some other fields are not included - `epilepsy_cause_categories` and `description` in multiaxial diagnosis are not included. The boolean fields describing features present or absent in a focal epilepsy are excluded - only focality (left or right) count to the total.

`number_of_completed_fields_in_related_models` performs this same process for all fields in related models. For example, a single multiaxial description of a child's epilepsy comprises multiple episodes, all of which are different.

This function steps through the audit questions one by one, calculating the minimum number of fields that can be expected to be completed, based on the selections the user has already made. As with the other functions it accepts an instance of the `Registration` model that relates to the child in question. It draws on two other helper functions:

- `scoreable_fields_for_model_class_name()`
- `count_episode_fields()`

These both do the same for related models if they exist. The `Episode` model is particularly complicated as it details all the different epilepsy types each of which have different decision trees that contribute to the scores.

5.8.7 Date Validations

Overview

Dates are sparsely gathered across E12 as these are generally annoying for users to enter as they are slower to enter than other fields and involve digging back through clinical notes to find. Where they are collected, it is because something is calculated from them.

FIRST PAEDIATRIC ASSESSMENT DATE

Orginally this was registration_date, but was subsequently refactored to become first_paediatric_assessment_date. This is set in the registration form and is a one way value. Once set, it cannot be undone, except by RCPCH staff. It is a critical date, since cohort allocation and end of first year of care are both calculated from it. It is also used in date validation as the earliest allowable date for some measures.

EPISODE DATE

In the multiaxial diagnosis form, a child may have several episodes of seizure, each one associated with a date. Information is collected on confidence around that date, since many seizures are unwitnessed. The earliest allowable date an episode can be given is the date of birth.

SYNDROME DIAGNOSIS DATE

In the multiaxial diagnosis form, a child may have more than one syndrome (though usually only one), each one associated with a date. The earliest allowable date a syndrome can be given is the date of birth.

COMORDIBIDITY DIAGNOSIS DATE

In the multiaxial diagnosis form, a child may have more than one comorbidity, each one associated with a date. The earliest allowable date a comorbidity can be given is the date of birth.

CONSULTANT PAEDIATRICIAN WITH EXPERTISE IN EPILEPSIES

2 dates are supplied and both are mandatory to complete the form - date referred and date seen The date seen cannot be before the date referred and the earliest allowable date is the date of birth.

CONSULTANT PAEDIATRIC NEUROLOGIST

2 dates are supplied and both are mandatory to complete the form - date referred and date seen The date seen cannot be before the date referred and the earliest allowable date is the first paediatric assessment date.

EPILEPSY SURGERY

Two dates are supplied if the child has been referred, but only the referral date is mandatory. The date seen cannot be before the date referred and the earliest allowable date is the first paediatric assessment date.

EPILEPSY NURSE SPECIALIST

2 dates are supplied and both are mandatory to complete the form - date referred and date seen The date seen cannot be before the date referred and the earliest allowable date is the date of birth (see issue 1007).

EEG

2 dates are supplied and both are mandatory to complete the form - date requested and date performed The date performed cannot be before the date requested and the earliest allowable date is the date of birth.

MRI

2 dates are supplied and both are mandatory to complete the form - date requested and date performed The date performed cannot be before the date requested and the earliest allowable date is the date of birth.

ANTIEPILEPSY MEDICINE

antiseizure medicine

2 dates are supplied - date discontinued and date started - but only date started is mandatory to complete the form if an antiseizure medicine has been given.

The date discontinued cannot be before the date started and the earliest allowable date is the date of first paediatric assessment.

rescue medicine

2 dates are supplied - date discontinued and date started - but only date started is mandatory to complete the form if an antiseizure medicine has been given.

The date discontinued cannot be before the date started and the earliest allowable date is the date of first paediatric assessment.

INDIVIDUALISED CARE PLAN DATE

One date is supplied. Earliest allowable date is the date of first paediatric assessment.

VALIDATION

Date validation occurs in `validators.py` and accepts a minimum of one date. If more than one date, a flag must be supplied to explain whether it is expected to be the earlier of the the two dates. The `earliest_allowable_date` parameter is an optional.

It raises a ValueError which is caught in the UI.

VALIDATION DATES SUMMARY TABLE

Model	Date	mandatory	earliest allowable date
Registration	first_paediatric_assessment_date	Yes	current submitting cohort start dat clinicians or date_of_birth if RCPC
Episode	seizure_onset_date	Yes	date_of_birth
Syndrome	syndrome_diagnosis_date	Yes	date_of_birth
Comorbidity	comorbidity_diagnosis_date	Yes	date_of_birth
Assessment	consultant_paediatrician_referral_date	Yes	date_of_birth
Assessment	consultant_paediatrician_input_date	Yes	consultant_paediatrician_referral_
Assessment	paediatric_neurologist_referral_date	Yes	first_paediatric_assessment_date
Assessment	paediatric_neurologist_input_date	Yes	paediatric_neurologist_referral_da
Assessment	childrens_epilepsy_surgical_service_referral_date	Yes	first_paediatric_assessment_date
Assessment	childrens_epilepsy_surgical_service_input_date	No	childrens_epilepsy_surgical_servic
Assessment	epilepsy_specialist_nurse_referral_date	Yes	first_paediatric_assessment_date
Assessment	epilepsy_specialist_nurse_input_date	Yes	epilepsy_specialist_nurse_referral_
Investigations	eeg_request_date	Yes	date_of_birth
Investigations	eeg_performed_date	Yes	eeg_request_date
Investigations	mri_brain_requested_date	Yes	date_of_birth
Investigations	mri_brain_reported_date	Yes	mri_brain_requested_date
AntiepilepsyMedicine	antiepilepsy_medicine_start_date	Yes	first_paediatric_assessment_date
AntiepilepsyMedicine	antiepilepsy_medicine_stop_date	No	antiepilepsy_medicine_start_date
AntiepilepsyMedicine	antiepilepsy_medicine_start_date	Yes	first_paediatric_assessment_date
AntiepilepsyMedicine	antiepilepsy_medicine_stop_date	No	antiepilepsy_medicine_start_date
Management	individualised_care_plan_date	Yes	first_paediatric_assessment_date

5.8.8 KPIs

The Key Performance Indicators are listed here

the `KPI` model stores these indicators as individual fields, together with the help and reference text that is signposted in the template.

```
class KPI(models.Model, HelpTextMixin):
    """
    Key performance indicator fields.

    The 12 key performance indicators, as specified by RCPCH, are:

    1. Paediatrician with expertise in epilepsies - % of children and young people with epilepsy, with input by a 'consultant paediatrician with expertise in epilepsies' within 2 weeks of initial referral

    2. Epilepsy Specialist Nurse - % of children and young people with epilepsy, with input by epilepsy specialist nurse within the first year of care

    3. Tertiary input - % of children and young people meeting defined criteria for paediatric neurology referral, with input of tertiary care and/or CESS referral within the first year of care

        3b. Epilepsy surgery referral - % of ongoing children and young people meeting defined epilepsy surgery referral criteria with evidence of epilepsy surgery referral

    4. ECG  - % of children and young people with convulsive seizures and epilepsy, with an ECG at first year

    5. MRI   - % of children and young people with defined indications for an MRI, who had timely MRI within 6 weeks of request

    6. Assessment of mental health issues  - %  of children and young people with epilepsy where there is documented evidence that they have been asked about mental health either through clinical screening, or a questionnaire/measure

    7. Mental health support - %  of children and young people with epilepsy and a mental health problem who have evidence of mental health support"

    8. Sodium Valproate - % of all females 12 years and above currently on valproate treatment with annual risk acknowledgement form completed

    9. (a) Comprehensive Care Planning agreement  - % of children and young people with epilepsy after 12 months where there is evidence of a comprehensive care plan that is agreed between the person, their family and/or carers and primary and secondary care providers, and the care plan has been updated where necessary

            9a. Patient held individualised epilepsy document/copy of clinic letter that includes care planning information - % of children and young people with epilepsy after 12 months that had an individualised epilepsy document with individualised epilepsy document or a copy clinic letter that includes care planning information

            9b. Patient/carer/parent agreement to the care planning - % of children and young people with epilepsy after 12 months where there was evidence of agreement between the person, their family and/or carers as appropriate

            9c. Care planning has been updated when necessary - % of children and young people with epilepsy after 12 months where there is evidence that the care plan has been updated where necessary

    10. (b) Comprehensive Care Planning content - % of children diagnosed with epilepsy with documented evidence of communication regarding core elements of care planning

            9a. Parental prolonged seizures care plan
                Percentage of children and young people with epilepsy who have been prescribed rescue medication and have evidence of a written prolonged seizures plan.

            9b. Water safety
                Percentage of children and young people with epilepsy with evidence of discussion regarding water safety.

            9c. First aid
                Percentage of children and young people with epilepsy with evidence of discussion regarding first aid.

            9d. General participation and risk
                Percentage of children and young people with epilepsy with evidence of discussion regarding general participation and risk.

            9e. SUDEP
                Percentage of children and young people with epilepsy with evidence of discussion regarding SUDEP and evidence of a prolonged seizures care plan.

            9f. Service contact details
                Percentage of children and young people with epilepsy with evidence of being given service contact details.

    11. School Individual Healthcare Plan - % of children and young people with epilepsy aged 4 years and above with evidence of a school individual healthcare plan by 1 year after first paediatric assessment..

    """

    """
    12. Percentage of children and young people with epilepsy, with input by a 'consultant paediatrician with expertise in epilepsies' within 2 weeks of initial referral

    Calculation Method
    Numerator = Number of children and young people  [diagnosed with epilepsy] at first year AND (who had [input from a paediatrician with expertise in epilepsy] OR a [input from a paediatric neurologist] within 2 weeks of initial referral. (initial referral to mean first paediatric assessment)
    Denominator = Number of and young people [diagnosed with epilepsy] at first year
    """
    paediatrician_with_expertise_in_epilepsies = models.IntegerField(
```

```
        help_text={
            "label": "1. Paediatrician with expertise in epilepsies",
            "reference": "Percentage of children and young people with epilepsy, with input by a 'consultant paediatrician with expertise in epilepsies'
within 2 weeks of initial referral",
        },
        default=None,
        null=True,
    )

    """
    13. Percentage of children and young people with epilepsy, with input by epilepsy specialist nurse within the first year of care.

    Calculation Method
    Numerator= Number of children and young people [diagnosed with epilepsy] AND who had [input from or referral to an Epilepsy Specialist Nurse] by first
year
    Denominator = Number of children and young people [diagnosed with epilepsy] at first year
    """
    epilepsy_specialist_nurse = models.IntegerField(
        help_text={
            "label": "2. Epilepsy Specialist Nurse",
            "reference": "Percentage of children and young people with epilepsy, with input by epilepsy specialist nurse within the first year of care.",
        },
        default=None,
        null=True,
    )

    """
    14. Percentage of children and young people meeting defined criteria for paediatric neurology referral, with input of tertiary care and/or CESS referral
within the first year of care.

    Calculation Method
    Numerator = Number of children ([less than 3 years old at first assessment] AND [diagnosed with epilepsy] OR (number of children and young people
diagnosed with epilepsy who had [3 or more maintenance AEDS] at first year) OR (Number of children less than 4 years old at first assessment with epilepsy
AND (generalised myoclonic seizures OR focal myoclonic seizures))  OR (number of children and young people diagnosed with epilepsy  who met [CESS criteria] )
AND had [evidence of referral or involvement of a paediatric neurologist] OR [evidence of referral or involvement of CESS]
    Denominator = Number of children [less than 3 years old at first assessment] AND [diagnosed with epilepsy] OR (number of children and young people
diagnosed with epilepsy who had [3 or more maintenance AEDS] at first year )OR (number of children and young people diagnosed with epilepsy  who met [CESS
criteria] OR (Number of children less than 4 years old at first assessment with epilepsy AND  (generalised myoclonic seizures OR focal myoclonic seizures))
    """
    tertiary_input = models.IntegerField(
        help_text={
            "label": "3. Tertiary input",
            "reference": "Percentage of children and young people meeting defined criteria for paediatric neurology referral, with input of tertiary care
and/or CESS referral within the first year of care.",
        },
        default=None,
        null=True,
    )

    """
    3b. Percentage of ongoing children and young people meeting defined epilepsy surgery referral criteria with evidence of epilepsy surgery referral.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy AND met [CESS criteria] at first year AND had [evidence of referral or
involvement of CESS]
    Denominator =Number of children and young people diagnosed with epilepsy AND met CESS criteria at first year
    """
    epilepsy_surgery_referral = models.IntegerField(
        help_text={
            "label": "3b. Epilepsy surgery referral",
            "reference": "Percentage of ongoing children and young people meeting defined epilepsy surgery referral criteria with evidence of epilepsy
surgery referral.",
        },
        default=None,
        null=True,
    )

    """
    15. Percentage of children and young people with convulsive seizures and epilepsy, with an ECG at first year.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND with convulsive episodes at first year AND who have [12 lead
ECG obtained]
    Denominator = Number of children and young people diagnosed with epilepsy at first year AND with convulsive episodes at first year
    """
    ecg = models.IntegerField(
        help_text={
            "label": "4. ECG",
            "reference": "Percentage of children and young people with convulsive seizures and epilepsy, with an ECG at first year.",
        },
        default=None,
        null=True,
    )

    """
    16. Percentage of children and young people with defined indications for an MRI, who had timely MRI within 6 weeks of request (KPI 5)

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND who are NOT (JME OR JAE OR CAE OR Generalised tonic clonic
seizures only OR self-limited epilepsy with centrotemporal spikes ~(SELECT)) AND who had an MRI within 6 weeks of referral.
    Denominator = Number of children and young people diagnosed with epilepsy at first year AND who are NOT (JME OR JAE OR CAE OR Generalised tonic clonic
seizures only OR self-limited epilepsy with centrotemporal spikes ~(SELECT))
    """
    mri = models.IntegerField(
```



```
        help_text={
            "label": "5. MRI",
            "reference": "Percentage of children and young people with defined indications for an MRI, who had timely MRI within 6 weeks of request",
        },
        default=None,
        null=True,
    )

    """
    17. Percentage of children and young people with epilepsy where there is documented evidence that they have been asked about mental health either through clinical screening, or a questionnaire/measure.

    Calculation Method
    Numerator = Number of children and young people over 5 years diagnosed with epilepsy AND who had documented evidence of enquiry or screening for their mental health
    Denominator = = Number of children and young people over 5 years diagnosed with epilepsy
    """
    assessment_of_mental_health_issues = models.IntegerField(
        help_text={
            "label": "6. Assessment of mental health issues",
            "reference": "Percentage of children and young people with epilepsy where there is documented evidence that they have been asked about mental health either through clinical screening, or a questionnaire/measure.",
        },
        default=None,
        null=True,
    )

    """
    18. Percentage of children and young people with epilepsy and a mental health problem who have evidence of mental health support

    Calculation Method
    Numerator =  Number of children and young people diagnosed with epilepsy AND had a mental health issue identified AND had evidence of mental health support received
    Denominator= Number of children and young people diagnosed with epilepsy AND had a mental health issue identified
    """
    mental_health_support = models.IntegerField(
        help_text={
            "label": "7. Mental health support",
            "reference": "Percentage of children and young people with epilepsy and a mental health problem who have evidence of mental health support",
        },
        default=None,
        null=True,
    )

    """
    19. Percentage of all females 12 years and above currently on valproate treatment with annual risk acknowledgement form completed

    Calculation Method
    Numerator = Number of females aged 12 and above diagnosed with epilepsy at first year AND on valproate AND annual risk acknowledgement forms completed AND pregnancy prevention programme in place
    Denominator = Number of females aged 12 and above diagnosed with epilepsy at first year AND on valproate
    """
    sodium_valproate = models.IntegerField(
        help_text={
            "label": "8. Sodium Valproate",
            "reference": "Percentage of all females 12 years and above currently on valproate treatment with annual risk acknowledgement form completed",
        },
        default=None,
        null=True,
    )

    """
    9A. Percentage of children and young people with epilepsy after 12 months where there is evidence of a comprehensive care plan that is agreed between the person, their family and/or carers and primary and secondary care providers, and the care plan has been updated where necessary.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND( with an individualised epilepsy document or copy clinic letter that includes care planning information )AND evidence of agreement AND care plan is up to date including elements where appropriate as below
    Denominator = Number of children and young people diagnosed with epilepsy at first year
    """
    comprehensive_care_planning_agreement = models.IntegerField(
        help_text={
            "label": "9A. Comprehensive care planning agreement",
            "reference": "Percentage of children and young people with epilepsy after 12 months where there is evidence of a comprehensive care plan that is agreed between the person, their family and/or carers and primary and secondary care providers, and the care plan has been updated where necessary.",
        },
        default=None,
        null=True,
    )

    """
    9i. Percentage of children and young people with epilepsy after 12 months that had an individualised epilepsy document with individualised epilepsy document or a copy clinic letter that includes care planning information.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND( with individualised epilepsy document or copy clinic letter that includes care planning information )
    Denominator = Number of children and young people diagnosed with epilepsy at first year
    """
    patient_held_individualised_epilepsy_document = models.IntegerField(
        help_text={
            "label": "i. Patient-held individualised epilepsy document/copy of clinic letter that includes care planning information",
            "reference": "Percentage of children and young people with epilepsy after 12 months that had an individualised epilepsy document with individualised epilepsy document or a copy clinic letter that includes care planning information.",
```



```
        },
        default=None,
        null=True,
    )

    """
    9ii. Percentage of children and young people with epilepsy after 12 months where there was evidence of agreement between the person, their family and/or
carers as appropriate.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND with evidence of agreement
    Denominator = Number of children and young people diagnosed with epilepsy at first year
    """
    patient_carer_parent_agreement_to_the_care_planning = models.IntegerField(
        help_text={
            "label": "ii. Patient/carer/parent agreement to the care planning",
            "reference": "Percentage of children and young people with epilepsy after 12 months where there was evidence of agreement between the person,
their family and/or carers as appropriate.",
        },
        default=None,
        null=True,
    )

    """
    9iii. Percentage of children and young people with epilepsy after 12 months where there is evidence that the care plan has been updated where necessary.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND with care plan which is updated where necessary
    Denominator = Number of children and young people diagnosed with epilepsy at first year
    """
    care_planning_has_been_updated_when_necessary = models.IntegerField(
        help_text={
            "label": "iii. Care planning has been updated when necessary",
            "reference": "Percentage of children and young people with epilepsy after 12 months where there is evidence that the care plan has been updated
where necessary.",
        },
        default=None,
        null=True,
    )

    """
    9B. Percentage of children diagnosed with epilepsy with documented evidence of communication regarding core elements of care planning.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND evidence of written prolonged seizures plan if prescribed
rescue medication AND evidence of discussion regarding water safety AND first aid AND participation and risk AND service contact details AND SUDEP
    Denominator = Number of children and young people diagnosed with epilepsy at first year
    """
    comprehensive_care_planning_content = models.IntegerField(
        help_text={
            "label": "9B. Comprehensive care planning content",
            "reference": "Percentage of children diagnosed with epilepsy with documented evidence of communication regarding core elements of care planning
(items a - f).",
        },
        default=None,
        null=True,
    )

    """
    9i. Percentage of children and young people with epilepsy who have been prescribed rescue medication and have evidence of a written prolonged seizures
plan.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND prescribed rescue medication AND evidence of a written
prolonged seizures plan
    Denominator = Number of children and young people diagnosed with epilepsy at first year AND prescribed rescue medication
    """
    parental_prolonged_seizures_care_plan = models.IntegerField(
        help_text={
            "label": "i. Parental prolonged seizures care plan",
            "reference": "Percentage of children and young people with epilepsy who have been prescribed rescue medication and have evidence of a written
prolonged seizures plan.",
        },
        default=None,
        null=True,
    )

    """
    9ii. Water Safety

    Percentage of children and young people with epilepsy with evidence of discussion regarding water safety.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND with evidence of discussion regarding water safety
    Denominator = Number of children and young people diagnosed with epilepsy at first year
    """
    water_safety = models.IntegerField(
        help_text={
            "label": "ii. Water safety",
            "reference": "Percentage of children and young people with epilepsy with evidence of discussion regarding water safety.",
        },
        default=None,
        null=True,
    )
```

```
    """
    9iii. First Aid

    Percentage of children and young people with epilepsy with evidence of discussion regarding first aid.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND with evidence of discussion regarding first aid
    Denominator = Number of children and young people diagnosed with epilepsy at first year
    """
    first_aid = models.IntegerField(
        help_text={
            "label": "iii. First aid",
            "reference": "Percentage of children and young people with epilepsy with evidence of discussion regarding first aid.",
        },
        default=None,
        null=True,
    )

    """
    9iv. General participation and risk

    Percentage of children and young people with epilepsy with evidence of discussion regarding general participation and risk.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND with evidence of discussion regarding general participation and
risk
    Denominator = Number of children and young people diagnosed with epilepsy at first year
    """
    general_participation_and_risk = models.IntegerField(
        help_text={
            "label": "iv. General participation and risk",
            "reference": "Percentage of children and young people with epilepsy with evidence of discussion regarding general participation and risk.",
        },
        default=None,
        null=True,
    )

    """
    9v. SUDEP

    Percentage of children and young people with epilepsy with evidence of discussion regarding SUDEP and evidence of a prolonged seizures care plan.

    Calculation Method
    Numerator = Number of children diagnosed with epilepsy AND had evidence of discussions regarding SUDEP
    Denominator = Number of children diagnosed with epilepsy at first year
    """
    sudep = models.IntegerField(
        help_text={
            "label": "v. Sudden unexpected death in epilepsy",
            "reference": "Percentage of children and young people with epilepsy with evidence of discussion regarding SUDEP (Sudden unexpected death in
epilepsy).",
        },
        default=None,
        null=True,
    )

    """
    9vi. Service contact details

    Percentage of children and young people with epilepsy with evidence of being given service contact details.

    Calculation Method
    Numerator = Number of children and young people diagnosed with epilepsy at first year AND with evidence of discussion of been given service contact
details
    Denominator = Number of children and young people diagnosed with epilepsy at first year
    """

    service_contact_details = models.IntegerField(
        help_text={
            "label": "vi. Service contact details",
            "reference": "Percentage of children and young people with epilepsy with evidence of being given service contact details.",
        },
        default=None,
        null=True,
    )

    """
    20. School Individual Healthcare Plan

    Percentage of children and young people with epilepsy aged 4 years and above with evidence of a school individual healthcare plan by 1 year after first
paediatric assessment.

    Calculation Method
    Numerator = Number of children and young people aged 4 years and above diagnosed with epilepsy at first year AND with evidence of EHCP
    Denominator =Number of children and young people aged 4 years and above diagnosed with epilepsy at first year
    """
    school_individual_healthcare_plan = models.IntegerField(
        help_text={
            "label": "10. School individualised health care plan",
            "reference": "Percentage of children and young people with epilepsy aged 4 years and above with evidence of a school individual healthcare plan
by 1 year after first paediatric assessment.",
        },
        default=None,
```



```
        null=True,
    )

    organisation = models.ForeignKey(
        "epilepsy12.Organisation", on_delete=models.CASCADE
    )

    parent_trust = models.CharField(max_length=250)

    class Meta:
        verbose_name = _("KPI ")
        verbose_name_plural = _("KPIs")

    def __str__(self):
        return f"KPI for child in {self.organisation.OrganisationName}({self.parent_trust})"
```

There is an instance of this model for each registration.

Scoring

Key performance indicators have 4 states:

- Failed (0)
- Passed (1)
- Ineligible (2)
- Unscored (None)

An example of an ineligible KPI would be a child with nonconvulsive epilepsy not needing an ECG.

This scoring system allows a child's individual score to be displayed clearly in the template using colours or icons to reflect their adherence to different measures, or for the scores to be aggregated together, for example to show how a give organisation performs against its peers in the same or another region. The results can be tabulated or mapped to show geographical variation, and sequentially against cohort to change over time.

The KPIs are final endpoint of the audit and therefore their accuracy is essential. A full suite of tests is in place to ensure this is true.

Note that the KPIs are only calculated for the **currently submitting** cohort that have completed a full year of care

The KPIs are aggregated to generate totals and percentages as well as averages across different levels of abstraction - by this is meant, either at organisational level, trust/health board level, or NHS region etc.

They are key part of the reporting dashboard.

5.8.9 Organisations, Trusts and Regions

Levels of Abstraction

The organisational structure of health care in England and Wales influences reporting and table structure.

ORGANISATIONS AND TRUSTS

This is the lowest level of abstraction and represents either an acute or a community hospital/organisation responsible for epilepsy care of children and young people. There are often several organisations in a Trust. Each organisation, like each Trust, has its own ODS code, and from year to year there is movement between trusts as organisations change their allegiances between trusts when mergers are carried out. The organisation model therefore has more than once instance for some organisations, as their parent status or other details such as name change.

On a monthly basis the NHS ODS API is polled with any changes to organisational structure, and the local database record for that organisation is updated to reflect the latest changes.

INTEGRATED CARE BOARDS

These were introduced in 2022 and superceded Clinical Commissioning Groups (CCGs) as the geographical commissioning areas within the NHS. The are 42 ICBs and trusts and their organisations fit neatly inside them like Russian dolls. Each ICB has its own ODS code and the ICB model is taken directly from NHS Digital with its boundary shapes for mapping, and is versioned. Currently there is no automated process to check for changes to boundaries or ICB membership and update the records. Note there are no ICBs in Wales.

LOCAL HEALTH BOARDS

These exist only in Wales and are both equivalent to Trust and ICB in England. One LHB might have several organisations and commissioning also is distributed across the 7 LHBs. As above, the model is taken from NHS Digital and includes the boundary shapes for mapping. Note there is no automated process currently to check for changes to boundaries or LHB membership and update the records.

OPENUK NETWORKS

These are networks of NHS Health Boards and Trusts that provide care for children with epilepsies, organised regionally and overseen by a UK Working Group. Not all centres are members of an OPEN UK network. There are no boundary shapes to describe each region, but each one has its own identifier, and therefore there is an entity model to hold information on each OPEN UK network referenced by each organisation.

NHS ENGLAND REGIONS

There are 7 of these in England and their model is taken from NHS Digital. Each one has its own boundary code. ICBs fit neatly inside each one.

LOCAL AUTHORITIES

Local authority codes for each organisation are not stored except for those organisations in London. Local authorities are administrative regions not related to health or the NHS. In London local authorities are usually referred to as London Boroughs. There is a boundary model for London Boroughs taken from NHS Digital and this is used only for mapping. As above, although versioned, there is no process in place to check for updates nationally and update the database record.

5.8.10 Reporting

Reporting is the principle reason for the Epilepsy12 Audit - to provide patients, their families and the teams caring for them meaningful information about the standard of their care. A secondary aim of the audit is to provide clinicians and hospital manager teams with real time information, filtered to their organisation, on their performance against the indicators, as well as other descriptors of their clinic - for example demographic information and case mix. Whilst it is not a clinical tool, it is meant to be used and updated in a clinical setting, and allow multidisciplinary teams to view the dashboard together at intervals and reflect on their service.

Filtering and aggregation

Each time a measure in the audit is scored for a given child, the KPIs are rescored for that child and stored in the KPI model.

For the purposes of reporting, a summary of all KPIs for a given cohort are needed at different 'levels of abstraction'. Users are interested not only in how the child's care has performed against the national standard, but also how all the children in their organisation and trust compare with the rest of their integrated care board, NHS England Region or local health board. The functions for first filtering all the children for a particular cohort and level of abstraction are found in `/common_view_functions/report_queries.py`. The KPI scores for these children are then aggregated in `/common_view_functions/aggregate_by.py`.

5.8.11 User Groups

The user groups are summarised here

Django allows permission-based and group based access. The user groups defined above are containers for permissions to all the models. Generic django permissions allow prescription of view, change, create and delete to each model (found in `epilepsy12/constants/user_types.py`).

```
# logged in user can view all national data but not logs
EPILEPSY12_AUDIT_TEAM_VIEW_ONLY = "epilepsy12_audit_team_view_only"

# logged in user can edit but not delete national data. Cannot view or edit logs or permissions.
EPILEPSY12_AUDIT_TEAM_EDIT_ACCESS = "epilepsy12_audit_team_edit_access"

# logged in user access all areas: can create/update/delete any audit data, logs, epilepsy key words and organisation trusts, groups and permissions
EPILEPSY12_AUDIT_TEAM_FULL_ACCESS = "epilepsy12_audit_team_full_access"

# logged in user can view all data relating to their trust(s) but not logs
TRUST_AUDIT_TEAM_VIEW_ONLY = "trust_audit_team_view_only"

# logged in user can edit but not delete all data relating to their trust(s) but not view or edit logs, epilepsy key words and organisation trusts, groups
and permissions
TRUST_AUDIT_TEAM_EDIT_ACCESS = "trust_audit_team_edit_access"

# logged in user can delete all data relating to their trust(s) but not view or edit logs, epilepsy key words and organisation trusts, groups and permissions
TRUST_AUDIT_TEAM_FULL_ACCESS = "trust_audit_team_full_access"

# logged in user can view their own audit data, consent to participation and remove that consent/opt out. Opting out would delete all data relating to them,
except the epilepsy12 unique identifier
PATIENT_ACCESS = "patient_access"
```

In addition, certain custom permissions associated with some models have been created:

```
# Case
CAN_LOCK_CHILD_CASE_DATA_FROM_EDITING = (
    "can_lock_child_case_data_from_editing",
    "Can lock a child's record from editing.",
)
CAN_UNLOCK_CHILD_CASE_DATA_FROM_EDITING = (
    "can_unlock_child_case_data_from_editing",
    "Can unlock a child's record from editing.",
)
CAN_OPT_OUT_CHILD_FROM_INCLUSION_IN_AUDIT = (
    "can_opt_out_child_from_inclusion_in_audit",
    "Can sanction an opt out from participating in the audit. Note all the child's date except Epilepsy12 unique identifier are irretrievably deleted.",
)

# Registration
CAN_APPROVE_ELIGIBILITY = (
    "can_approve_eligibility",
    "Can approve eligibility for Epilepsy12.",
)
CAN_REMOVE_APPROVAL_OF_ELIGIBILITY = (
    "can_remove_approval_of_eligibility",
    "Can remove approval of eligibiltiy for Epilepsy12.",
)

CAN_REGISTER_CHILD_IN_EPILEPSY12 = (
    "can_register_child_in_epilepsy12",
    "Can register child in Epilepsy12. (A cohort number is automatically allocaeted)",
)
CAN_UNREGISTER_CHILD_IN_EPILEPSY12 = (
    "can_unregister_child_in_epilepsy12",
    "Can unregister a child in Epilepsy. Their record and previously entered data is untouched.",
)

CAN_ALLOCATE_EPILEPSY12_LEAD_CENTRE = (
    "can_allocate_epilepsy12_lead_centre",
    "Can allocate this child to any Epilepsy12 centre.",
)

CAN_TRANSFER_EPILEPSY12_LEAD_CENTRE = (
    "can_transfer_epilepsy12_lead_centre",
    "Can transfer this child to another Epilepsy12 centre.",
)

CAN_EDIT_EPILEPSY12_LEAD_CENTRE = (
    "can_edit_epilepsy12_lead_centre",
    "Can edit this child's current Epilepsy12 lead centre.",
)

CAN_DELETE_EPILEPSY12_LEAD_CENTRE = (
    "can_delete_epilepsy12_lead_centre",
    "Can delete Epilepsy12 lead centre.",
```



```
)

CAN_CONSENT_TO_AUDIT_PARTICIPATION = (
    "can_consent_to_audit_participation",
    "Can consent to participating in Epilepsy12.",
)
```

These permissions can be access in the view or in the template to constrain access to particular fields, views or models.

Logging

A security requirement was the ability to track user activity. This is done in a number of ways.

LOGIN SIGNPOST

```
@receiver(user_logged_in)
def log_user_login(sender, request, user, **kwargs):
    logger.info(f'{user} ({user.email}) logged in from {get_client_ip(request)}.')
    VisitActivity.objects.create(
        activity=1,
        ip_address=get_client_ip(request),
        epilepsy12user=user
```

This picks up a successful login and reports this back in the template and stores the time and IP address. This code is taken from `signals.py`.

USER LOGGING

This pulls the same login and logout data from the `VisitActivity` and reports it in a table, accessible from the user table, depending on level of permission access.

AUDIT TRAIL

django-simple-history is a library dependency that creates a history model for each model, prepended by 'history_'. It is possible then through the django admin interface to view all models and model fields and identify which users have touched them.

5.8.12 Users

Epilepsy12 has subclassed the BaseUserManager for Django. This maintains certain core Django features relating to password hashing and storage, but allows some customisation. The Epilepsy12 user therefore does not use 'username' as its primary identifier, rather it uses email.

User creation

Signup is not user driven, and there is no capacity for account creation from the login page. Instead, lead clinicians are created by RCPCH audit team members through a user work flow accessible from the organisation landing page. It is possible, depending on permissions, from this point to create users for individual organisations, or RCPCH audit team members who have broader access but are not affiliated with any organisation. There are additionally a small number of clinicians who also are part of the RCPCH Epilepsy12 team and therefore have national access.

Depending on permission, users can view all users in a given Trust/Local Health Board, and allocate users accordingly. A user is created from a standard Django form (`epilepsy12/forms_folder/epilepsy12_user.py`), where role, name and email are entered. On submit, validation occurs in the standard Django way, and if the form `is_valid` , this generates an email sent to the new user, who clicks on the individualised link (valid for 72 hours) to activate the account.

User management is controlled from the user table, by those with appropriate permissions. From the user table it is possible to:

1. add new users
2. edit user details or groups
3. delete users
4. see activity logs

PASSWORD RESET

This workflow is available from the login page and is the standard Django workflow, with css styling to personalise the forms (`templates/registration`) to match Epilepsy12 and RCPCH design guidelines.

RESEND PASSWORD

If the user has not clicked on the link within 72 hours, or the email has been lost in junk, the administrator can send a further email from the newly created user form. The workflow is identical to user creation.

5.8.13 Views

For design reasons explained elsewhere, function-based views were preferred over class-based views. This means that each field in a given model is updated in isolation of others through individual ajax post requests direct from the template.

Decorators

Decorators are used to protect the views.

- `@login_required` : Nearly all routes are decorated by the django decorator and redirect to the login page
- `@user_may_view_this_organisation()` : The is a custom decorator which only allows access to the logged in user whose employing organisation (as stored in the request.user object) matches that of the lead Epilepsy centre for the child. Superusers or RCPCH members may see all children nationally. Failure redirects to 403
- `@user_may_view_this_child()` : This is a custom decorator which allows only the logged-in user access to view or edit data relating to a given child in their same organisation. Failure redirects to 403
- `user_can_access_user` : This is a custom decorator which allows only the logged-in user access to view or edit data relating to a given user in their same organisation. Failure redirects to 403
- `@permission_required("epilepsy12.change_episode", raise_exception=True)`

View structure

View functions mirror the structure of the models. Each model has a corresponding view in the `views` folder in the root of the `epilepsy12` folder. The first function in each file is called to load the form template. If no instance of that model exists, an instance is created. Any dependencies for the template (such as lists for select dropdowns etc) are retrieved here and added to the context to be passed on to the template.

VIEW FUNCTIONS

Any function that updates, creates or deletes a model, early on in the function, calls `validate_and_update_model` in `epilepsy12/common_view_functions/` . It accepts the following parameters:

- `request` : request object passed in from calling view
- `model_id` : id of the model to update
- `model` : the Model itself
- `field_name` : the name of the field to be updated as string
- `page_element` : the type of page element selector, one of `date_field` , `hospitals_select` , `multiple_choice_multiple_toggle_button` , `select` , `single_choice_multiple_toggle_button` , `toggle_button` , `snomed_select`
- `comparison_date_field_name=None` : if the selector is a `date_field` , additional parameters are required for validation
- `is_earliest_date=None` : if the selector is a `date_field` , additional parameters are required for validation
- `earliest_allowable_date=None` : if the selector is a `date_field` , additional parameters are required for validation

For the final 3 optional parameters, see the section on date validation in `validators.py`

Any validations that identify errors are raised here with messages that are caught in the view. Otherwise the model is updated with the new value.

The name of the view function matches the name of the field of the model affected. It is passed also in the url.

Example

```
@login_required
@user_may_view_this_child()
@permission_required("epilepsy12.change_episode", raise_exception=True)
def seizure_onset_date(request, episode_id):
    """
    HTMX post request from episode.html partial on date change
    """
```



```
    try:
        episode = Episode.objects.get(pk=episode_id)
        error_message = None
        validate_and_update_model(
            request=request,
            model=Episode,
            model_id=episode_id,
            field_name="seizure_onset_date",
            page_element="date_field",
            earliest_allowable_date=None,  # episodes may precede the first assessment date or cohort date
        )
    except ValueError as error:
        error_message = error

    keywords = Keyword.objects.all()
    episode = Episode.objects.get(pk=episode_id)

    context = {
        "episode": episode,
        "seizure_onset_date_confidence_selection": DATE_ACCURACY,
        "episode_definition_selection": EPISODE_DEFINITION,
        "keyword_selection": keywords,
        "epilepsy_or_nonepilepsy_status_choices": sorted(
            EPILEPSY_DIAGNOSIS_STATUS, key=itemgetter(1)
        ),
        "epileptic_seizure_onset_types": sorted(
            EPILEPSY_SEIZURE_TYPE, key=itemgetter(1)
        ),
        "GENERALISED_SEIZURE_TYPE": sorted(GENERALISED_SEIZURE_TYPE, key=itemgetter(1)),
        "LATERALITY": LATERALITY,
        "FOCAL_EPILEPSY_MOTOR_MANIFESTATIONS": FOCAL_EPILEPSY_MOTOR_MANIFESTATIONS,
        "FOCAL_EPILEPSY_NONMOTOR_MANIFESTATIONS": FOCAL_EPILEPSY_NONMOTOR_MANIFESTATIONS,
        "FOCAL_EPILEPSY_EEG_MANIFESTATIONS": FOCAL_EPILEPSY_EEG_MANIFESTATIONS,
        "nonepilepsy_onset_types": NON_EPILEPSY_SEIZURE_ONSET,
        "nonepilepsy_types": sorted(NON_EPILEPSY_SEIZURE_TYPE, key=itemgetter(1)),
        "syncopes": sorted(NON_EPILEPTIC_SYNCOPES, key=itemgetter(1)),
        "behavioural": sorted(
            NON_EPILEPSY_BEHAVIOURAL_ARREST_SYMPTOMS, key=itemgetter(1)
        ),
        "sleep": sorted(NON_EPILEPSY_SLEEP_RELATED_SYMPTOMS, key=itemgetter(1)),
        "paroxysms": sorted(NON_EPILEPSY_PAROXYSMS, key=itemgetter(1)),
        "migraines": sorted(MIGRAINES, key=itemgetter(1)),
        "nonepilepsy_miscellaneous": sorted(EPIS_MISC, key=itemgetter(1)),
        "epilepsy_cause_selection": EPILEPSY_CAUSES,
    }

    response = recalculate_form_generate_response(
        model_instance=episode.multiaxial_diagnosis,
        request=request,
        template="epilepsy12/partials/multiaxial_diagnosis/episode.html",
        context=context,
        error_message=error_message,
    )

    return response
```

This function is called from the following path in `urls.py`, as an HTMX POST request from the template, called in the change event of the custom page element `date_field`.

```
path(
        "episode/<int:episode_id>/seizure_onset_date",
        views.seizure_onset_date,
        name="seizure_onset_date",
    ),
```

`seizure_onset_date` is a field in the `Episode` model. The request contains the `episode_id` and the new date in the header. It can be accessed by `request.POST.get(request.htmx.trigger_name)`.

After the episode to be updated with the new date has been retrieved using the the `episode_id`, these parameters are passed into `validate_and_update_model` where the date is retrieved, validated and the model updated. If there are any validation errors, these are raised here, and caught in the `try...except` block and stored in the `error_message` variable.

The `context` is updated in the `recalculate_form_generate_response` discussed elsewhere before being passed back to the template. This latter function also calculates the number of scored fields in the form and updates the totals in the `steps.html` partial by adding an HTMX custom trigger to the header.

5.9 API

Further information to on the Epilepsy12 API to follow.

The API exists to allow implementers of Epilepsy12 to autopopulate some fields in the Epilepsy12 audit with data from their own systems such as their EPR, PAS or other systems.

At present none of our end user trusts have indicated they would like to use this feature. If you are interested in using the API please contact us.

5.10 Environment Variables

5.10.1 Adding a new environment variable - checklist

When adding a new environment variable (env var), please follow these steps:

- Choose the env var name carefully. It should be unambiguous, and should not be a common name that might be used by other software. It should be in `UPPERCASE`, with `UNDERSCORES_SEPARATING_WORDS`. It should be descriptive, and should not be an abbreviation unless the abbreviation is widely understood.
- Add the new env var to the **non-committed** `envs/.env` file. This file should be used to set the environment variables for your local development. **This file should not be committed to the repository**. You can use real values for the environment variables in this file. **Group related ENVs together** in this file, following the style of the `env-template` file.
- Add the new env var to the `envs/env-template` file, which is the ONLY file in the /envs/ folder that should be committed to the repository. In this file you can set a **default/suggested value** for the variable (assuming that there is no security risk to disclosure of this), and add a **comment** to explain what it does. Commit and push this file to the repository. Group related env vars together in this file, keeping it the same order as the `.env` file.
- Usually the env var's value will be accessed somewhere in `settings.py` with something like `os.getenv("ENV_VAR_NAME")`. Try to keep the env var name consistent with the variable name used in the code. Also aim to **group related settings together** in the settings file. Comments here are useful in knowing what the variable does and aid maintainability.
- Make the env var available to GitHub Actions and the deployment environments. See below for details.

Pushing the env vars to GitHub for Actions to use it

There is a script in the `s/` folder which uses the `gh` GitHub CLI tool to push the contents of your `envs/.env` file as a base64-encoded string to a single GitHub secret called ENVIRONMENT in the E12 repository. This is then used by the GitHub Actions to decode and supply the appropriate env vars to the software stack during automated tests. You **can** safely push confidential information to this secret, as it is only accessible to the E12 repository.

Pushing the env vars to our deployment environments

At present the env vars are manually copied into `/var/epilepsy12/envs/.env` in each of the deployment environments - Development, Staging and Live. This is done by the system admin over SSH. It is possible that in future we will use a more automated system for this.

Sharing the env vars with other developers

All developers who have a local development environment will to need to be informed that there is a new env var, what an acceptable value fo it is, and what it does. One way to inform the team is using our secure Signal chat. Generally, confidential env var values are not shared by email or other insecure methods.

Further documentation

Detailed documentation about the environment variable is **only** required if it not immediately clear from the name and comments that it does and how it is used.

5.11 Deployment

This page is a work in progress, as we are currently working on migrating to this new deployment method.

Set up a new VPS server instance and access it via SSH

Adds a command to make OS updates quicker and more consistent

```
sudo echo "alias doupdates='sudo apt update && sudo apt dist-upgrade && sudo apt autoremove && sudo apt autoclean'" >> /etc/profile;source /etc/profile
```

Run the new `doupdates` command

```
doupdates
```

Automatically `cd` to the E12 folder when you log in

```
echo "cd /var/epilepsy12/" >> /etc/profile;source /etc/profile
```

Add other SSH users using their public keys from GitHub

```
ssh-import-id gh:eatyourpeas
ssh-import-id gh:pacharanero
```

Install Docker

```
curl -fsSL get.docker.com | bash
```


Optional: Install Zsh and Oh-my-zsh

Install Zsh and Oh-my-zsh

Two tools I find quite nice when spending a lot of time in the terminal are Zsh and oh-my-zsh. These give you additional features in the terminal, such as better tab completion and a terminal prompt which includes the Git branch you are currently using.

Note that if you escalate your privileges to `root` using `sudo su` or similar then the new root shell will not have Zsh set up, so you may need to install Zsh/Oh-my-zsh for both your user **and** the root user.

Install Zsh

```
sudo apt install zsh
```

Install oh-my-zsh

```
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

Optional: Change the ZSH_THEME from the default to my preferred "bira"

```
sed -i '/ZSH_THEME="robbyrussell"/c\ZSH_THEME="bira"' ~/.zshrc
```

Optional: Add some useful plugins for Git, Docker etc

```
sed -i '/plugins=(git)/c\plugins=(git git-extras wd docker docker-compose)' ~/.zshrc
```

You can also edit `~/.zshrc` to customise

```
nano ~/.zshrc
```

Clone the Epilepsy12 repository

This command clones the repo into the `/var/epilepsy12` folder, which we have chosen as the standard location for the E12 application.

```
git clone https://github.com/rcpch/rcpch-audit-engine.git /var/epilepsy12
```

`cd` into the new folder

```
cd /var/epilepsy12/
```

`git checkout` the branch you want to deploy

development **staging** **live**

```
git checkout development

git checkout staging

git checkout live
```

Useful commands

Outputs all the environment variables from within Docker Compose to the console, this can be useful for debugging deployments.

```
docker compose config
```


5.12 Documentation

5.12.1 Introduction

The RCPCH Audit Engine / Epilepsy12 documentation site is made with Material for MkDocs, which is a framework, separate from Django, which takes Markdown source files from `documentation/docs` within the project and compiles them into a static HTML site. These static files are then served from our hosting resources.

5.12.2 Docker development setup

As part of our standard Docker and Docker Compose development setup, we have a `docker-compose.yml` file in the root of the repo which will build a `mkdocs` Docker image with all the dependencies needed to run the documentation site locally.

By default this image is running in a container at `localhost:8001` when you run the Docker dev setup using `s/up` and it will auto-reload when you make changes to the source files in `documentation/docs`.

> **IMPORTANT**
>
> There are two ways to view the documentation site:
>
> https://localhost:8001 is served from the `mkdocs serve` command and **has auto-reload**, so is much more convenient for local development, you can instantly see changes made to the Markdown files in `documentation/docs`.
>
> https://e12.localhost/docs is the built static HTML files, served through Caddy. This reflects how the documentation site works in the VPS environments. **There is no auto-reload**, so you have to manually run `docker compose restart mkdocs` to rebuild the static HTML files after making changes to the Markdown files in `documentation/docs`.

5.12.3 How to edit content

- Generally any significant changes will need to be on a new Git branch, which by convention we name according to the 'slugified' title of the Issue that the changes resolve. Occasionally we will make small changes directly on the `development` branch, but this is not recommended.
- Make changes to the Markdown files in the `documentation/docs` folder.
- Ensure any new or renamed files are listed in the `nav` data structure within `mkdocs.yml` or they won't show up in the navigation.
- Save and review the auto-reloaded site on https://localhost:8001.
- 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, and makes reviewing PRs easier.

5.12.4 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 chasing unpredictable and cascading 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 in-browser, 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 in a browser, and will sync the changes with GitHub for you.

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 Developer 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 for MkDocs theme together add 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 our 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

5.12.5 Pull requests (no commit rights)

If you are not a member of the RCPCH Developer team, you many not have commit rights to the documentation repository, so to publish your changes you will need to submit a pull request. This is a standard GitHub process, but if you are not familiar with it, here are the steps:

- Push your changes to a branch on **your** fork of the repository.
- The branch should ideally be name according to the feature or fix it includes.
- Go to your fork of the repository on GitHub and click the 'Pull Request' button.
- Submit a pull request from your fork to the `development` branch of the main repository.

5.12.6 Deployment (for RCPCH Developer team)

See Deployment for details of how the documentation site is deployed.

5.12.7 NOT RECOMMENDED: Setting up a Python and Pyenv development environment for the E12 documentation site

Use the Dockerised development environment if you can

If you cannot use the Dockerised development environment, then you will need to set up a Python environment on your local machine to run the documentation site locally. We highly recommend using the Dockerised development environment if you can, as it is much easier to set up and use.

Create a virtualenv for the Python modules:

- Install `pyenv` using the instructions at https://github.com/pyenv/pyenv-installer
- Any recent Python version works, we tend to use 3.11
- Calling it `mkdocs` will enable Pyenv to automatically select it when you navigate to the directory, because this will match the contents of the `.python-version` file in the root of the project.

```
pyenv virtualenv 3.11 mkdocs
```

The first time you want to use the `mkdocs` pyenv, you will need to activate it. Subsequent times it should automatically be activated if you have named it the same as the entry in the `.python-version` file in the root of the project.

```
pyenv activate mkdocs
```

Install Material for Mkdocs

Install all the Python requirements

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

Running the development MkDocs server

`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`.

To run `mkdocs serve` the fastest way, use

```
export ENABLE_PDF_EXPORT=0;mkdocs serve  --config-file documentation/mkdocs.yml
```

`ENABLE_PDF_EXPORT=0 disables the generation of the PDF version of the documentation, which is slow and not needed in development.

`--config-file documentation/mkdocs.yml` tells MkDocs to use the `mkdocs.yml` file in the `documentation` folder. This assumes that you are running the command from the root of the project.

If you want the automatic PDF generation to happen in development locally, then run

```
export ENABLE_PDF_EXPORT=1;mkdocs serve  --config-file documentation/mkdocs.yml
```

The PDF generation slows down the hot reloading by about 10-15 seconds so it can get tiresome in development. PDF generation will automatically happen in production when the site is built and deployed, even if you didn't generate PDFs in local development.

6. Contact

6.1 Contact Page

For enquiries please contact the project team:

Email: sitecontactemail@example.com
Tel: 020 7092 6157 / 6056

You can find more information about the audit at https://www.rcpch.ac.uk/epilepsy12.

7. Legal

7.1 Intellectual Property

7.1.1 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.

7.1.2 Open Source Licenses

Software

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

Documentation and textual outputs

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

Upstream components

Our licensing arrangements do not affect the licenses of any upstream technologies which we have used to build pur platforms, such as Django (3-clause BSD) MkDocs (BSD) or Python (PSF). However our license choice is compatible with these upstream licensing arrangements.

7.2 License (CC-BY-SA 4.0)

License for the textual and documentation portions of this work

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.

7.2.1 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.

7.2.2 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.

7.3 Clinical Safety

7.3.1 Clinical Safety

The Epilepsy12 audit engine does not have any role in the direct clinical management of patients, it is solely for recording the milestones in childrens' care for the purposes of national audit, which is a secondary use of data, not a direct care use. It does not therefore require a clinical safety case or a Clinical Safety Officer.

7.3.2 Medical Device Registration

The Epilepsy12 audit engine does not have any role in the clinical management of patients, it is solely for recording the milestones in childrens' care for the purposes of national audit. It is not a medical device and as such requires no medical device registration.

7.4 Privacy Overview

Within the Epilepsy12 project we take the privacy of patients and our duty of confidentiality very seriously. We have complied with all relevant legislation and moral and ethical codes to support the project.

7.4.1 National Data Opt Out Implementation (England only)

The Secretary of State for Health and Social Care, having considered the advice from the Health Research Authority Confidentiality Advisory Group, have granted Epilepsy12 an exemption to the National Data Opt-Out (NDO). Epilepsy12 were granted this exemption because applying the National Opt-Out would introduce biases to the data and make it difficult to monitor care safety and quality at Trust level, risking quality of care and patient safety.

This means that healthcare providers in England no longer need to screen patients against the opt-out list prior to entering their data into Epilepsy12.

Patients can still opt out of Epilepsy12 by contacting their Epilepsy clinical team

Withdrawal of consent can be indicated in the audit detail submission for that patient, and the record will be deleted. Please take a look at the clinician user guide for more information on this.

Withdrawal of consent can be indicated in the audit detail submission for that patient, and the record will be deleted. Please take a look at the clinician user guide for more information on this.

7.4.2 Privacy Notice

The Privacy Notice for the Epilepsy12 Project is viewable or downloadable from the Privacy Notice page

7.4.3 Data Protection Impact Assessment

Our Data Protection Impact Assessment is viewable or downloadable from the DPIA page

7.5 Privacy Notice

7.5.1 Privacy Notice

The RCPCH has published a Privacy Notice for the Epilepsy 12 Project, these are embedded below in English and Welsh. You can also download them from the below links:

Privacy Notice (English) | Privacy Notice (Welsh)

Privacy Notice (English)

If the PDF cannot be embedded here, it is available to download from this link

Privacy Notice (Welsh)

If the PDF cannot be embedded here, it is available to download from this link

7.6 Data Protection Impact Assessment

The RCPCH's Data Protection Impact Assessment for the Epilepsy12 Audit is below. You can also download it from this link.

Data Protection Impact Assessment

If the PDF cannot be embedded here, it is available to download from this link

7.7 Section 251 Exemption

The Epilepsy12 audit engine has been granted an exemption from Section 251 of the NHS Act 2006 by the Confidentiality Advisory Group (CAG) of the Health Research Authority (HRA). This means that healthcare providers in England no longer need to screen patients against the opt-out list prior to entering their data into Epilepsy12.

We are reviewed annual for compliance with conditions of the exemption, and a selection of the most recent and relevant documents are embedded below. You can see more information on the Epilepsy12 section of the RCPCH website

Section 251 Support – Annual Review 2023

If the PDF cannot be embedded here, it is available to download from this link

Section 251 - Change in Data Processor to use RCPCH and sub-processors

If the PDF cannot be embedded here, it is available to download from this link

NDO Deferral Request – Conditionally Supported

If the PDF cannot be embedded here, it is available to download from this link

Section 251 Support – Annual Review 2022

If the PDF cannot be embedded here, it is available to download from this link

7.8 Terms of Service

Important

The Terms of Service must be accepted as a condition of use of the Epilepsy12 Platform

7.8.1 Key points

- You will maintain a secure, confidential password and assume responsibility for all activities that occur under your user account.
- The Royal College of Paediatrics and Child Health (RCPCH) makes no express or implied warranties with regard to the content or reports from the site.
- The Epilepsy12 platform is not an electronic medical record for direct care purposes and should not be used as such. You will refer to the official patient medical record when making medical decisions.

7.8.2 Acceptance of Terms

The RCPCH ("we", "us" or "our") provides access (the "Site") subject to your acceptance of these Terms. These Terms may be updated by the RCPCH from time to time without prior notice. By accessing, browsing, entering data or otherwise accessing the Site, you become a User and assert that you have read and understand and agree to be bound by the terms of this Agreement. If you have any questions about this Disclaimer, please contact us.

7.8.3 Your Account Obligations

You are responsible for maintaining a secure, confidential password, and for all activities that occur under your user account. You agree to notify us immediately of any unauthorised use of your username or password or any other breach of security.

7.8.4 Disclaimers

The medical data and other contents of the Site (“Contents”) were furnished by and entered into the RCPCH site by third parties with the permission of the RCPCH and are presented for informational purposes only. Content contained within or reported by the RCPCH does not supersede the official patient medical record.

The RCPCH makes no representations, warranties, or assurances as to the accuracy, currency or completeness of the Content provided. You accept the responsibility to verify any contents obtained from the site. Contents should be verified with the official patient medical record before making medical decisions. You should not reply on anything contained on this site to suggest a course of treatment for any medical condition.

Contents do not substitute for review of an individual’s medical record and consultation with a physical or other qualified health provider. Clinics in England should apply the NHS National Data Opt Out process and not submit data from patients who have opted out.

7.8.5 No warranties

The RCPCH does not make any express or implied warranties, presentations, or endorsements whatsoever (including, but not limited to, warranties of title or non-infringement, or the implied warranties of merchantability or fitness for a particular purpose) with regard to the site, the content, any reports provided through the site, and the RCPCH shall not be liable for any cost or damage arising either directly or indirectly from any content or use of the site.

It is solely your responsibility to evaluate the reliability, accuracy, timeliness, completeness or usefulness of all services and content provided through the site. The RCPCH does not warrant that the site will be uninterrupted or error-free or that defect in the site will be corrected. The site and the content made available on the site are provided on an ‘as is’ and ‘as available’ basis.

7.8.6 Fair Processing Statement

The RCPCH has HSCA Section 251 approval for Epilepsy12 to collect patient identifiable data without explicit patient consent (see references below). This data is processed and reported by the Epilepsy12 project team within the RCPCH with the Healthcare Quality Improvement Partnership (HQIP) as the Data Controller.

Full details of fair processing documentation for the collection of patient identifiable information without consent in Health Boards and Trusts in England and Wales will be added to this page prior to the start of the clinical audit data entry phase. Copies of these materials will also be provided to participating Health Boards and Trusts to display in clinic areas and share with their patients and family members. HQIP may approve the sharing of pseudonymised Epilepsy12 data for the purpose of service improvement if stringent data protection policies and arrangements can be demonstrated by requestees and the aims of the service improvement are approved.

Epilepsy12 data may also be linked to data from the Hospital Episode Statistics (HES), Patient Episode Database for Wales (PEDW) and the Office for National Statistics (ONS). NHS Digital and the NHS Wales Informatics Service may provide HES, PEDW and ONS data to Epilepsy12 in an anonymous format. The linked data would then be analysed by the Epilepsy12 project team at the RCPCH to further help to measure standards of care.