School of Electronic Engineering
and Computer Science

Final Report

Programme of study:
BSc Computer Science

Project Title:
Automation of the Royal College of Radiologists' Job Description and Advisory Appointment Committee Processes

Supervisor:
Dr Fabrizio Smeraldi

Student Name:
Christian Juresh

Final Year
Undergraduate Project 2023/24

Date: April 25, 2024

Acknowledgements

I want to thank Dr. Fabrizio Smeraldi, my supervisor, for his guidance and support throughout my project.

I am also grateful to all staff at the Royal College of Radiologists, particularly to Lucy Horder for bringing me on board the Professional Services team, and to Donna Barry for her training on Job Descriptions and Advisory Appointment Committees.

Most importantly, I'd like to thank my family, friends, and partner for their constant love and support throughout my entire degree.

Abstract

The Royal College of Radiologists (RCR) is a professional body responsible for advancing radiology and oncology throughout most of the United Kingdom. This includes oversight of high standards in the recruitment of Specialty and Associate Specialist (SAS) Radiologists, Consultant Radiologists, SAS Oncologists, and Consultant Oncologists across over 100 NHS Trusts, and some Crown Dependencies. This process includes mandatory statutory requirements.

Before a Trust advertises a position, they submit the Job Description (JD) to the RCR for review to ensure that the post is well-supported and adequately funded. The Advisory Appointment Committee (AAC), a panel of interviewers, evaluates candidates for these roles. The RCR contributes a Representative to the panel to verify that the candidates' training aligns with the role's requirements and to identify any additional training needs.

The current system depends heavily on manual operations, requiring significant involvement from multiple teams. This project aims to develop an automated system that will reduce time commitments, and decrease the likelihood of errors.

Contents

Chapter 1: Introduction

1.1 Background

Ex Radiis Salutas
From Rays, Health

The Royal College of Radiologists (RCR) plays a pivotal role in ensuring high standards in the recruitment process within radiology and oncology. Of particular importance is the quality assurance provided by Job Description (JD) review and Advisory Appointment Committee (AAC) panels. The RCR states that its processes ensure "candidates have clarity and confidence in applications, recruiting organisations attract and retain the best possible appointee, and vacancies are filled efficiently, benefitting existing consultant staff." (The Royal College of Radiologists 2022)

The National Health Service (NHS) understands the complexity of this process, and states that "The administrative burden associated with job planning is considerable. Success depends on having systems in place and information available", and Trusts are highly recommended to invest in electronic job planning software (The National Health Service 2017).

This is of particular importance, given the concerning shortfall in clinical radiology and oncology. With shortfalls of 17% in oncology and 29% in radiology, safe and effective care is limited. Moreover, these shortfalls are expected to worsen in the future. Doctors have had their pay cut, and 83% report some form of burnout (The Royal College of Radiologists 2023). This has profound implications for patient care and service delivery. Limb (2022) found that shortages lead to longer wait times for diagnoses and treatments, potentially worsening patient outcomes, in fact every month of delayed cancer treatment increases patients' risk of death by 10%. By 2029, these professions will need to grow by 45 percent (The National Health Service 2018).

Annual job plans, as defined by the RCR, are agreements between doctors and their employers "setting out the duties, responsibilities and objectives of the doctor" (Clinical radiology job planning guidance for consultant and SAS doctors 2022, 2022). This process, which involves completing review forms in Microsoft Word and subsequent email communication, often leads to back-and-forth discussions between the NHS Trust, RCR, and Regional Speciality Advisor (RSA). Every detail, including every date an email is sent or received, is meticulously recorded in a large Excel spreadsheet. This process ensures that job plans are fair and realistic.

On the other hand, the AAC process is "a legally constituted interview panel established by an employing body when appointing consultants." (www.rcseng.ac.uk, n.d.). Representatives are mandated by the NHS to ensure that the panel consti-

tutes a balanced Committee (The National Health Service (Appointment of Consultants) Regulations Good Practice Guidance, 2005). This process, involving the handling of sensitive data such as doctors' emails and phone numbers, requires encryption and password protection. Trusts often require multiple lists to find an available representative, as these individuals, being consultant doctors themselves, typically have busy schedules. The RCR's involvement ensures that the selection process adheres to high professional standards and that candidates are evaluated fairly and competently.

Their process is deliberately extensive for good reason. Generic healthcare recruitment software does not address the specialised, nuanced steps that are required by the RCR. Many aspects are legal requirements by the NHS that must be followed by every non-foundation Trust and Royal College (www.rcseng.ac.uk, n.d.). Automation would allow for the streamlining of these operations, significantly reducing the administrative burden and likelihood of manual errors. Offering a more reliable and consistent approach to managing the recruitment process is a necessity to cope with the increasing demands of healthcare delivery, and to support the overburdened workforce.

1.2 Problem Statement

Despite its critical role in maintaining employment standards, the RCR's current system faces significant inefficiencies and limitations. Managed primarily through manual operations involving Excel, Outlook, and Word, the risk of errors is high, contributing to an already time-consuming and labour-intensive process. These stand-alone applications have limited integration capabilities, requiring manual data transfer, and making automation of repetitive tasks extremely difficult, unreliable, or downright impossible. Although Excel is powerful for data manipulation, it is ill-suited for tracking complex workflows. It lacks the ability to monitor process stages, complex data integrity mechanisms, concurrency and multi-user environments, and handling of large datasets. Consequently, the RCR is forced to create a new spreadsheet annually, increasing time wasted searching through multiple different files. The lack of data normalisation and continuity makes it difficult to track and analyse specialities, which is essential for reporting.

Data analysis is critical for improving efficiency. The RCR states that Advisory Appointment Committee (AAC) data is collected to "check that the appointee is qualified to train doctors for the future, track increasing or decreasing numbers of doctors, track increases or decreases in different types of posts, track where it may be difficult for NHS Trusts to attract new recruits" (www.rcr.ac.uk, n.d.). Without these metrics it becomes difficult to provide guidance on how to allocate resources correctly, which is essential for resolving consultant shortages.

Delays and errors, inherent in manual systems, can cause Trusts to miss critical deadlines for filling vacancies and to lose out on high-quality candidates. This must be avoided at all costs, as the NHS Appointment of Consultants Regulations state

that "Only in extreme circumstances should it be necessary to cancel an AAC." (The National Health Service (Appointment of Consultants) Regulations Good Practice Guidance, 2005). Significant staff time, which could be better utilised in more critical roles, is dedicated to this tedious and manual process. This issue affects not only the RCR, but also Trusts' teams, Regional Speciality Advisors (RSAs) reviewing Job Plans, RCR representatives assessing panels, and the candidates who are pivotal in delivering lifesaving patient care.

1.3 Aim

This project aims to develop an automated system specifically tailored for the Royal College of Radiologists (RCR), to manage and accelerate their Job Plan and Advisory Appointment Committee processes. Integrating an SQL database will ensure that a more robust system supports an automated workflow, works with large datasets and complex data relationships, and ensures data integrity and security. This integration will give the RCR the ability to analyse the data for reporting with business analytics services such as PowerBI. Choosing a web-based platform application allows for easier access and collaboration across different users and departments. A stand-alone application would be extremely unpopular, as all users will have to install a separate application, which is also very likely to be blocked by IT systems. A website also allows users to access the site from any device. Web development frameworks such as SvelteKit and Django are selected not only because they streamline development tasks but, but also because it is much easier to automate complex business workflows with the tools they provide. Implementing data security and compliance with encryption software like OpenSSL, password protection, and secure data storage solutions will allow the project to comply with GDPR and NHS regulations. By creating a user-friendly, customised, and accessible interface, we can simplify the process for all stakeholders.

Chapter 2: Literature Review

2.1 Current Process

2.1.1 Job Descriptions

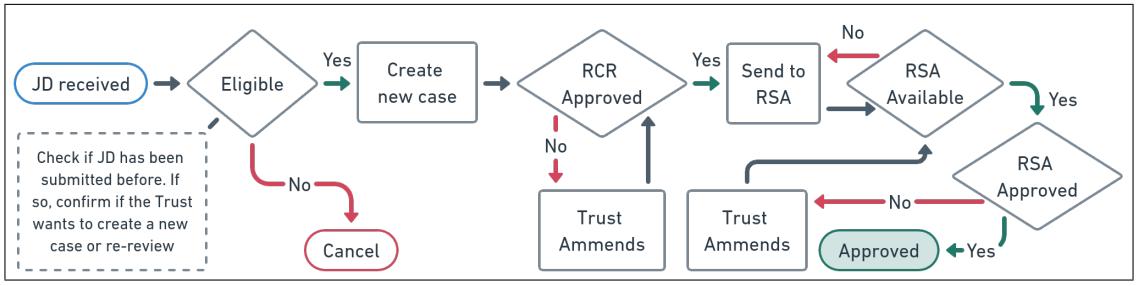


Figure 2.1: JD Process

The Job Description (JD) process in figure 2.1 reflects the meticulous approach followed by the Royal College of Radiologists (RCR) to ensure quality and precision. The process begins with the receipt of a job description, which is then subjected to preliminary checks. If the JD meets the initial criteria, it moves forward to a more detailed examination by a Regional Speciality Advisor (RSA). If the RSA is unavailable at any stage, a new RSA is promptly assigned. The diagram shows multiple checkpoints and potential loops that can occur if information is missing or incomplete, yet every effort must be made to complete this entire process within 2 weeks. The process concludes by informing the Trust that they are permitted to advertise their post and by sending them a form to complete to request a list of representatives.

2.1.2 Advisory Appointment Committees

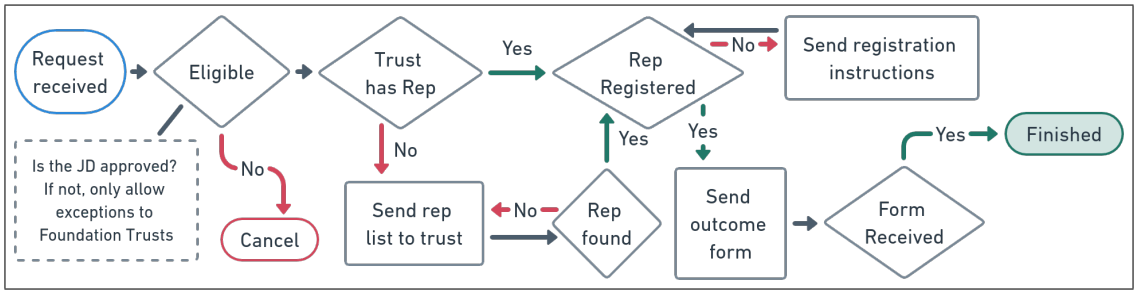


Figure 2.2: AAC Process

The Advisory Appointment Committee (AAC) process in figure 2.2 is equally rigorous and focuses on giving Trusts the best chance of securing a representative. This process begins by checking whether the previous Job Description (JD) process was successful, only granting exceptions when requested by a Foundation Trust. Like the JD process, it often involves loops when sending multiple lists to the Trust. If the Trust opts to source their own representative, either upon request or after an unsuccessful search, the appropriate guidance and rules must be sent. All representatives must complete training and registration to attend panels. The process concludes when the representative returns an outcome form sent to them to complete after the panel is held, or if the Trust cancels the panel for any reason.

2.2 Related Works

2.2.1 Existing Storage Systems

With Excel, the primary limitation is the lack of ability to store speciality data. This is because job posts often have multiple primary specialities and secondary subspecialities. If opting for a single column, data integrity becomes almost impossible as Excel does not inherently support complex data validation for multiple entries within a single cell. This makes it difficult to ensure that the data entered is consistent and makes analysing and querying this data challenging. Using multiple columns for each specialisation provides a more structured way to store this data. However, in this context it quickly becomes unwieldy – in our case we are left with 44 columns. Here the spreadsheet becomes cluttered and difficult to navigate.

The same issues apply to tracking the date of each step of the process, as due to loops in the workflow we often encounter multiple dates in each cell. With no automation whatsoever, these dates, as well as the workflow's status, must be entered manually every time.

The core issue with both approaches is the lack of normalisation. By splitting data into multiple related tables, and establishing relationships between them, we can reduce redundancy and improve data integrity. This is only realistic if we use a relational database management system like SQL.

2.2.2 Healthcare Recruitment Tools

Searching for software to assist with recruitment in the healthcare sector will bring up results such as ICIMS and The Access Group. Unfortunately, these programs are typically directed towards the Trust rather than Royal Colleges. Their intended user base is employers looking to advertise and interview candidates, not for those involved in the rigorous job plan review and representative selection process required by the Royal College of Radiologists (RCR). These solutions typically require private meetings, so understanding their capabilities is difficult. For a set of requirements this specific, only personalised software is suitable.

2.2.3 Other Royal Colleges

Finding specific details on the software used by other Royal Colleges in the UK is not possible as this information is not typically disclosed publicly. Though the general process for Advisory Appointment Committees (AAC's) is somewhat standardised across various Royal Colleges due to the strict requirements set out by the NHS, there is no central piece of software. The advantages to an open-source solution may span beyond just the Royal College of Radiologists (RCR).

While the overall process is shared, there are still many differences between colleges. Each email and form would differ, and doctors in various departments have distinct needs. A software flexible enough to accommodate everyone is far beyond the scope of this project. For example, the Royal College of Physicians (RCP) re-

quires that medical staffing departments complete the job description review form themselves, in contrast to the RCR filling it out for them. This may, however, be an indication that this project should accommodate any policy changes that the RCR may implement. The RCP also requires foundation Trusts to submit representative requests with a minimum of eight weeks' notice, a guideline that is optional but recommended in many other colleges, and offers an optional kitemark (RCP London, 2017). Some colleges require the trust to complete the review form for themselves and the reviewer. This is an example of an optimisation to the workflow that could be implemented into the system.

With differing requirements, it's no wonder that simply sharing software with each other is not a solution. A project that would suit every Royal College would be an expensive undertaking.

Chapter 3: Design

3.1 Visuals

The interface will utilize a dual-theme approach: a light mode with a subtle dotted background to enhance readability during daytime, while a dark mode will reduce eye strain for those working later hours. With more employees choosing flexible working options due to the shift to working from home, it is important for users to reduce the amount of blue light in the evening. The primary colour, purple, aligns with the Royal College of Radiologists' (RCR's) branding, creating a professional and cohesive aesthetic. Red will be used selectively for critical actions like logout and reject buttons, drawing immediate attention.

Clean, sans-serif fonts will be used for easy legibility, and consistent layout structures will be maintained, even across different screens, to support muscle memory and navigability. The layout will take some inspiration from the recent trend of Bento UI, where information is broken into separate cards of a single category. This design style is ideal for a project that requires the users to have access to so much data, because it allows for compartmentalising information effectively. 'isms' such as Neumorphism and Glassmorphism will be used sparingly. Whilst they add depth to the design, allowing users to differentiate between elements we want them to focus on most, they can also be distracting, and appear unprofessional when overused.

3.2 Frameworks

3.2.1 Django

Django is chosen for its comprehensive feature set that simplifies many web development tasks thanks to its "batteries-included" design. Django's ORM (Object-Relational Mapping) system will be used to manage database operations much more efficiently than through direct SQL, which is crucial given the complex data relationships in the RCR's processes. Its built-in security features as less essential as it will have limited internet access, however tools such as SQL Injection protection and Cross-Site Scripting (XSS) protection are still paramount.

To turn Django into an API, Django-Ninja will be implemented for its exceptional speed and ease of use. The use of Pydantic with Django-Ninja ensures strong type checking, significantly reducing the risk of bugs that are usually common in Python and JavaScript. Its integration with OpenAPIs automatic documentation allows us to test the API in SwaggerUI before writing any frontend code, and can even be used to transfer schemas from backend to frontend. This library also supports Django's asynchronous capabilities. However, it will not be used in this project as the frequency of requests is not high enough to warrant the additional complexity, though it leaves room for scalability if needed.

Django-Ninja-JWT, built on top of Django-Ninja-Extra, provides JWT (JSON Web Tokens) authentication, essential for secure API access in our application. This feature allows for stateless authentication, making it easier to manage sessions in a distributed environment where the RCR's data may need to be accessed securely across multiple locations. The JWTs will be stored securely using HTTP-only cookies, configured with strict properties to prevent unauthorized access. It also eliminates the need to handle CSRF tokens in Django, though SvelteKit already handles this by default.

PyTransitions

The Python 'transitions' library for state management of JDs. State machines are crucial for enforcing the rules and transitions of a workflow's various stages, and ensure that each job description progresses through its lifecycle in a controlled, traceable manner, preventing errors and ensuring consistency. It also gives users the ability to track the progress of each JD visually by generating state machine diagrams based on the current state in relation to others.

3.2.2 SvelteKit

You may have noticed that there is no dedicated Backend and Frontend section. This is because SvelteKit in itself is a full-stack framework. There are a number of reasons for this, most notably that the frontend, Svelte, and backend are extremely closely integrated and are essentially one entity. This allows us to have features such as a file based routing system, simplified state management, and much more. This is a key advantage of SvelteKit over other frameworks, as it allows for a more seamless development experience. The performance is also superb, as it compiles to vanilla JavaScript and removes the overhead of the virtual DOM - though the performance difference with a framework such as Vue3 is negligible. On low-end devices or poor network connections however, which can be common where organisations are underfunded, adaptive loading and progressive enhancement can make a huge difference. As discussed in 1.1, is extremely important that this system is as accessible as possible, especially to Trusts that need it most, as a large part of the National Health Service depends on this process.

The use of a 'second backend' allows a more modular design. By allowing Django to focus exclusively on serving and managing the database, SvelteKit's backend can focus on rendering the frontend, handling user interactions, and manipulating form data into a cleaner format. This leaves Svelte to handle the UI with data in a format easiest for HTML to display.

Libraries

TypeScript will allow strict typing, just like in the Django-Ninja Schema. As discussed in 3.2.1, OpenAPI-Typescript allows us to import this schema into SvelteKit, providing type safety and autocompletion without having to rewrite all of the types

in a completely different language. This ensures that the frontend and backend development environment are always in sync, reducing the likelihood of errors and inconsistencies.

These types will be used in requests with Axios, which will be responsible for communicating with Django. It is easier to work with than native JavaScript's Fetch as it is more readable with less boilerplate, but more importantly, it has automatic JSON transformation which will be essential for handling the complicated forms that will be built. A custom API library will be built on top to abstract this code even more, which can turn over 10 lines of code into a single function call. These functions can be used anywhere in the program, and since the same data is likely to be used in multiple different pages, it also removes the need to repeat code. Lastly, by having a single source of truth for the API, not only is it easier to make changes to the API without having to change every single page that uses it, but it also reduces the changes of bugs due to typos to almost zero.

Postmark is an email delivery service which will be used for email verification and notifications. It is preferred over self-hosted emailing, as those are usually blocked as spam by all major email services, but still allows us to use our own custom domains through DKIM and Return-Path DNS records. It features a very easy to use API, and a very high deliverability rate, as all developers have to be approved by Postmark through a manually reviewed form.

shadcn-svelte is not so much a library, but rather a set of re-usable components build on top of a number of other libraries. It allows us to keep a consistent design across the entire application, and reduce time spent on building basic parts of the UI. This does however mean that the multitude of libraries it is built on top of have to be learnt and a deep understanding of the documentation is a requirement. These libraries will include; Svelte Headless Table for data tables. Formsnap, Superforms and Zod for form processing and validation. Bits UI (built on top of Melt UI) for the majority of the components. And TailwindCSS for styling and positioning. The form libraries are particularly important. While forms are one of the common features on the web, they are also one of the most complex - and this project is almost entirely based on large, complicated, and dynamic forms.

Luckily this setup will allow us to build well-structured and semantically correct forms, with client and server side validation. There are also Accessible Rich Internet Application (ARIA) attributes and proper labels for screen readers to make the website accessible to all users. Lastly they are generally easy to use and can be navigated exclusively with a keyboard.

3.3 Deployment

For the purposes of this project, the website will be deployed on a Linux server with Nginx, Gunicorn, and Node with its respective SvelteKit adaptor. Nginx will be used as the reverse proxy and HTTP server, along with Cerbot for automatic SSL

certificates. The Django admin panel may also be exposed publicly to allow access for RCR Employees working from home. Gunicorn will be used with multiple workers to handle multiple requests at once, as opposed to using django-ninja's asynchronous features. Node.js for SSR will improve load times and SEO. Though the latter is not as important, assuming the website will be a link in an existing RCR webpage, it will still help users searching for the website avoid multiple navigations. This may reduce the number of phone calls and emails received attempting to find the correct RCR department.

3.4 File Structure

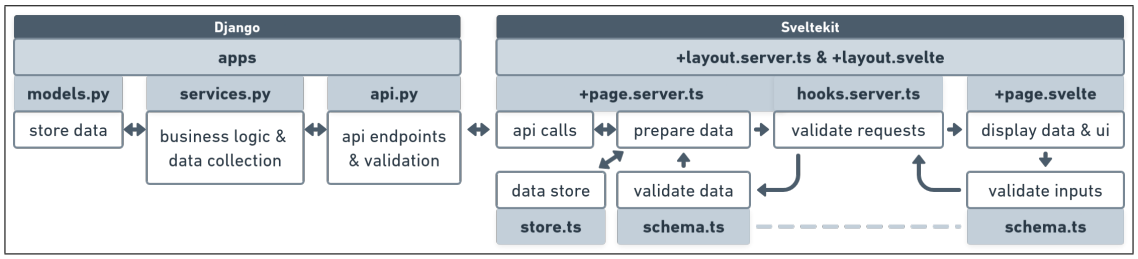


Figure 3.1: A high level representation of the code's file structure

The diagram in figure 3.1 presents the general communication between files, starting with `models.py` which defines and creates the database. `services.py` is used as an intermediary between the database and the API, and is responsible for handling the business logic. This includes complex algorithms to refine and filter data before moving it - which are usually called as functions in the api. `api.py` is then used to communicate with the frontend. The code here should be as simple and readable as possible, and rely entirely on functions defined in `services.py`. This is because the majority of the processing code is reusable, and it is best to separate concerns by keeping the API code to a number of function calls with understandable names, some error handling, user validation, and returns.

These files will be repeated in six different apps to separate concerns, improve modularity, maintainability, and scalability. These will include users, roles, trusts, specialities, jds, and aacs. Communication between these apps will be handled through `services.py` and occasionally `api.py`. Thanks to django-ninjas routers, it will also give each of our API endpoints a url that begins with the app name, making it easier to understand. Most of the apps are self-explanatory, however, to be clear the roles app will be used to define the type of employee to make it easier to handle their permissions (this includes Reviewers, Representatives, and the RCR and Trust teams). The full structure of this database will be discussed in 4.1.

`+layout.server.ts` is a server-side script that acts as a data pre-processer for all child routes below it. It will fetch data that is shared across all pages, mostly for the navigation bar. `+layout.svelte` is the global layout of the application presented to the user. In our case, it will hold the navigation bar, define the structure of the page by providing a wrapper for other pages, and render a small logo in the corner as the footer. `+page.server.ts` is the server-side script that prepares the

data which will be used on the corresponding page. It will fetch data from the API, potentially perform server-side processing, and then pass the data to the frontend to be rendered. `hooks.server.ts` is a server-side hook that will be solely responsible for validating requests over the API by checking whether the user is authenticated and authorised. `+page.svelte` is responsible for rendering the user interface. It will display data to the users and handle their interactions. Form data will often be passed back to `+page.server.ts`' Actions, where it will be validated server side, potentially processed, and then sent back to django through an API call. `schema.ts` is a single file per page that defines the form schemas for that route. It is first sent through `+page.server.ts` to `+page.svelte`, where it is used to validate the form data live as the user makes inputs. It is used again in the backend to validate the data before it is processed, this is crucial because users could easily bypass the frontend validation.

3.5 Routes

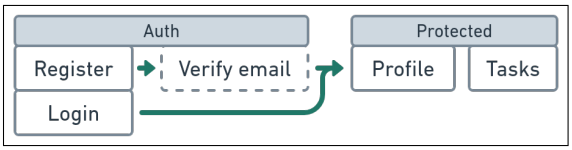


Figure 3.2: A high level representation of the navigation flow for all users

Figure 3.2 represents the general navigation we want all users to take. The grey cells represent the parent route, which will appear in the url, and the white cells may be a component, or child route. The reason for a shared auth route is that we can have one page have a login and register form without a page redirection. Verify email will actually be a child route with the sole purpose of verifying and redirecting the user.

The protected route is where all other pages will be. SvelteKits server hook will require all users to be authenticated through the Django API before they can access any pages - the most basic security measure. A common decision is to put authentication inside `+layout.server.ts`, as it encompasses all other routes after all. However, this is a massive security risk, as even unauthenticated uses can access the data fetched from the database. This is because the layout load function runs in parallel with the page's load function, so even if an unauthenticated user is redirected, the layout may still fetch data from the database. This isn't the only risk, for example if a user logs out in one tab, but keeps another tab open on a page that requires authentication, refreshing the second tab would bypass the authentication check because the layout's load function would not be aware that the user is no longer logged in.

All users will be initially redirected to the Profile page, which is to encourage keeping accurate records as they will see them every time the log-in. These records are important as they are automatically sent alongside forms that the user would otherwise have to fill out every time. They also have a Tasks button, which will automatically list all JDs and AACs that require action in one place.

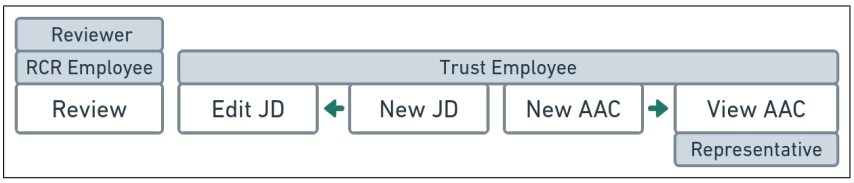


Figure 3.3: A high level representation of the navigation flow for specific users

Figure 3.3 represents the navigation flow for specific users. The white cells represent buttons in the nav-bar. The process technically starts at different stages for all users, however they all depend on a JD. A Trust Employee will start by clicking the New JD button in the nav-bar, and is redirected to that page. Once submitted, they are automatically redirected to the Edit JD page, which also has its own button. This page will display information about the JD, and provide a form for the Trust Employee to complete. This form will include a dynamic amount of questions, dependent on the consultant type of the JD, and answer fields for content such as page numbers and general comments. Navigation to the Edit JD page through a button will present a list of still editable JDs specific to that user which can be edited at any time.

Once saved, the next step is to be completed by the RCR Employee, who will have the same page as the Trust Employee, but with an added row for comments. Ofcourse, just like Edit JD, they are first presented with a list of JDs that require review. The RCR Employee will be able to save their changes, and submit or reject when they are ready. When submitting, they will be given a list of Reviewers they can send the JD to with relevant information, in order to help chose the best option. Reviewers will have the same page, but again with an added row for comments.

Once a JD is approved, the Trust Employee will be able to click the New AAC link to create a new panel. They will be able to select as many JDs as they like from a list, along with the date of the panel. They will then be redirected accordingly to View AAC. This page will have a list of valid Representatives along with contact details, in order of suitability. The Trust can choose one Representative, who will be given access to that AAC. Either the Trust or Representative can then download and upload an Outcome Form, which will be sent to the RCR Employee for data analysis. The reason this step is not a form is that interview outcomes are sometimes written by hand, or in person, where a Doctor may prefer not to use a computer.

Chapter 4: Implementation

4.1 Database

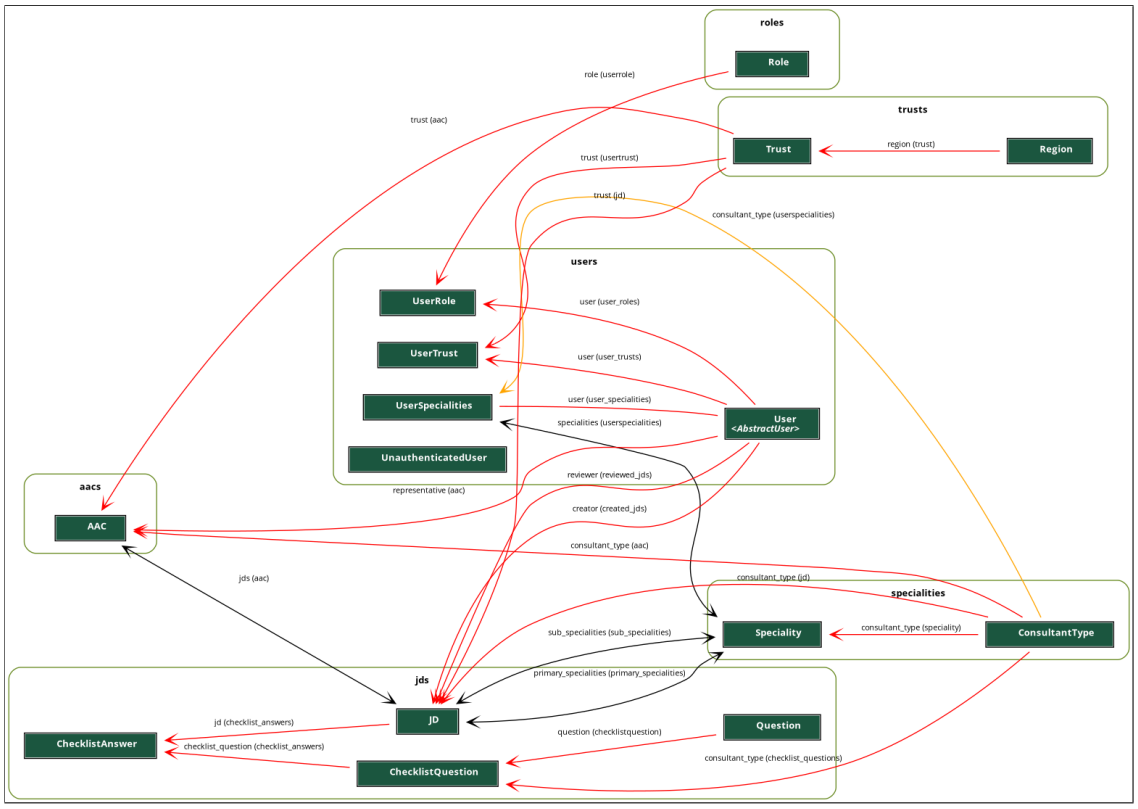


Figure 4.1: The Database Schema without fields

Figure 4.1 represents the database schema. Due to the large number of models, fields have been omitted in this figure to maintain readability. See Appendix A for a full version. This diagram aptly presents just how intertwined this system is, as well as the sheet number of data taken into consideration during the process.

This relational database schema allows for handing of detailed user profiles, Job Descriptions (JDs), and Advisory Appointment Committees (AACs). It allows for a sophisticated access control system based on roles, Trust, and consultant types. The abundant use of foreign keys for relationships shows the normalisation practices taken to avoid data redundancy.

This schema allows handling detailed user profiles, job descriptions, and associated evaluations. It allows for a sophisticated access control system based on roles, trust levels, and consultant types, all of which can be tailored to the specific needs of an organization or system. The database schema utilizes foreign keys for relationships, suggesting it's a relational database, and employs typical normalization practices to avoid data redundancy.

Normalization: The database avoids redundancy by normalizing data. Normalization helps in organizing data efficiently and reduces the redundancy by ensuring that each data item is stored only once. For instance, separating user details from roles and trust levels allows for easy updates and management of each aspect without unnecessary duplication. Clarity in Relationships: The clear definition of relationships between entities, like one-to-many and many-to-many, indicates thoughtful consideration of how different data points relate to each other. This clarity supports accurate and efficient querying of the database. Use of Abstract Entities: The use of an AbstractUser entity suggests an object-oriented approach to database design. This can make it easier to extend the database in the future because common attributes are centralized in the abstract entity. Flexibility in Role Management: The separation of UserRole, UserTrust, and UserSpecialities allows for a flexible permission and access control system. It can handle complex scenarios where users can have multiple roles, varying levels of trust, and multiple specialties. Hierarchical Specialization: The categorization of specialties into primary and sub-specialties indicates a hierarchical approach. This is beneficial for complex systems where specialties need to be organized in a detailed and hierarchical manner. Modularity: The schema seems to be modular, as indicated by the presence of separate entities like aacs and jds. This modularity allows for distinct parts of the application to evolve independently and makes the system easier to maintain. Data Integrity: The schema likely enforces data integrity through the use of foreign keys which ensure that relationships between entities are consistent and that orphan records are avoided. Comprehensive Data Capture: Entities have attributes that capture a broad range of data points, from user information to job descriptions, which would enable detailed analysis and reporting. Scalability: With its structured approach to data separation and relationships, the schema allows for scalability. As the system grows and the amount of data increases, such a schema can handle the growth without major changes. Security Considerations: By separating UnauthenticatedUser and User, the schema indicates a focus on security, allowing the system to differentiate between verified and unverified users. Regional Specificity: The inclusion of Region in the Trust entity allows the database to cater to regional variations in data, which can be crucial for businesses operating in multiple jurisdictions. Timestamps for Tracking Changes: Entities like User and JD contain timestamps, which are essential for tracking changes over time and for audit purposes.

4.2 app.pcss

4.3 REDO THIS

4.4 Django

4.4.1 Database

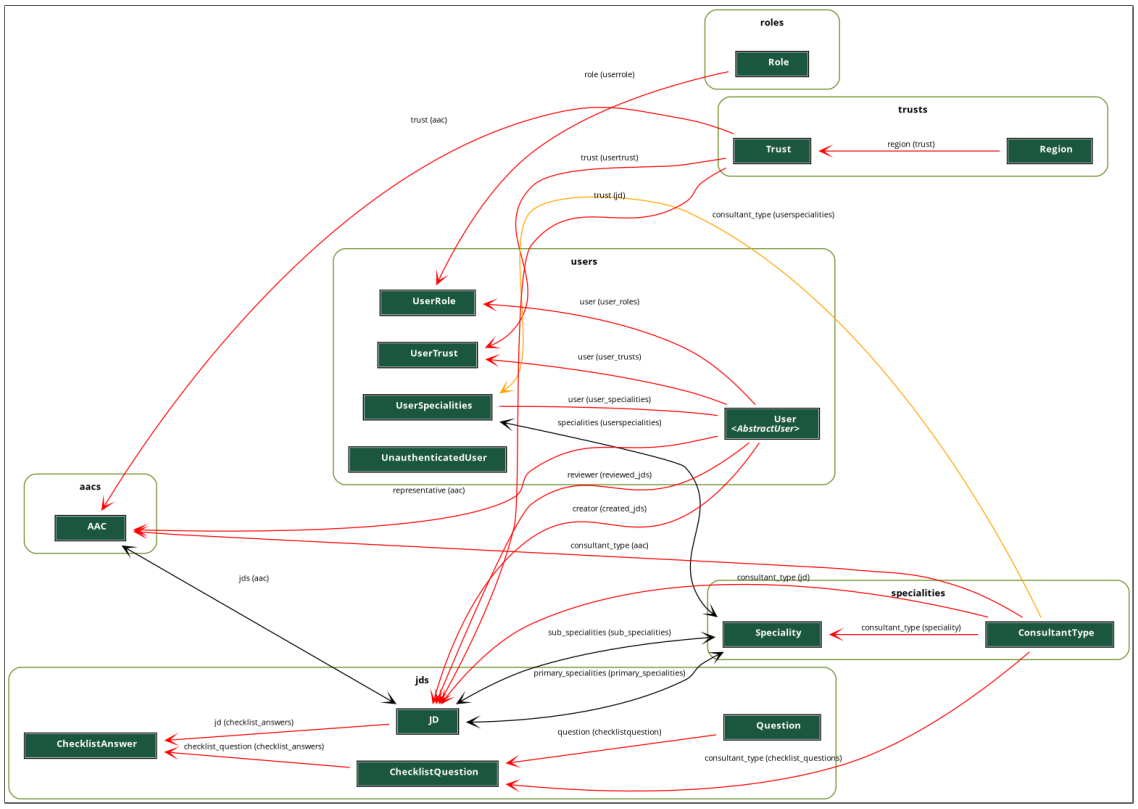


Figure x represents the database used. Due to the number of models, fields have been omitted in this version to maintain readability. See figure x in appendix x for a full version.

4.4.2 Schemas

pydantic

4.4.3 Services

django

4.4.4 API Endpoints

django-ninja & django-ninja-jwt

4.5 SvelteKit

4.5.1 Design

The front page greets you with a dark/light mode and a background with dots and a login auth page with

It is perfectly reactive and works on all devices and zoom levels. This is essential to accessibilty, and to allow doctors and trusts to make changes on the go.

By adding shadows to the main content we want the user to focus on, we can accelerate the rate at which they can learn the system. Components with no shadows are used for information that isn't as relvant, but may still be useful on occasions.

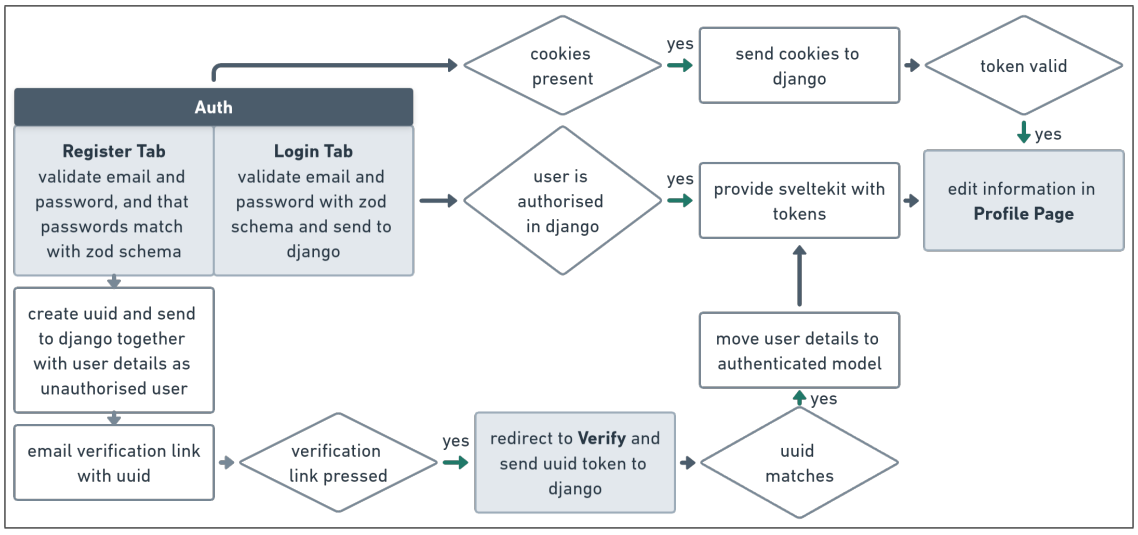
4.5.2 API Consumption

4.5.3 Authentication

Good security is essential in this process, as any bad actors could access sensitive information such as personal details, as well as cause havoc on the system. There is potential for one person to delay the radiology and oncology in the uk. Therefore

We do not need CSRF tokens as ...

As seen in Figure x, all routes past the protected/ route go through the hook (which is), which checks if the user is authenticated. If they are not, they are redirected to the login page. This is done by checking the session cookie, which is set when the user logs in. This is a secure way of checking if the user is authenticated, as the session cookie is encrypted and cannot be tampered with. JWT.



4.5.4 Profile

4.5.5 Panel

This page is incomplete. This is where the list of JDs and AACs will be displayed.

References

Limb, Matthew (June 10, 2022). "Shortages of radiology and oncology staff putting cancer patients at risk, college warns". In: *BMJ*, o1430. ISSN: 1756-1833. DOI: 10.1136/bmj.o1430. URL: https://www.bmj.com/lookup/doi/10.1136/bmj.o1430 (visited on 04/24/2024).

Royal College of Surgeons of England (2024). *Advisory Appointment Committees (AACs)*. Royal College of Surgeons. URL: https://www.rcseng.ac.uk/standards-and-research/support-for-surgeons-and-services/aacs/ (visited on 04/24/2024).

The National Health Service (2017). *Consultant job planning: a best practice guide.* URL: https://www.england.nhs.uk/wp-content/uploads/2022/05/consultant-job-planning-best-practice-guidance.pdf..

— (2018). *Strategic Framework for Cancer Workforce 05-07*. URL: https://www.hee.nhs.uk/sites/default/files/documents/Cancer-Workforce-Document_FINAL%20for%20web.pdf.

The National Health Service (Appointment of Consultants) The National Health Service (Appointment of Consultants) (2005). *The National Health Service (Appointment of Consultants) Regulations Good Practice Guidance*. URL: https://www.rcseng.ac.uk/-/media/Files/RCS/Standards-and-research/Support-for-surgeons/NHS_AAC20GoodPractice_Guidance.pdf.

The Royal College of Physicians of London (2017). *RCP London Advisory Appointments Committees (AAC)*. URL: https://www.rcplondon.ac.uk/education-practice/advice/advisory-appointments-committees-aac#:~:text=.

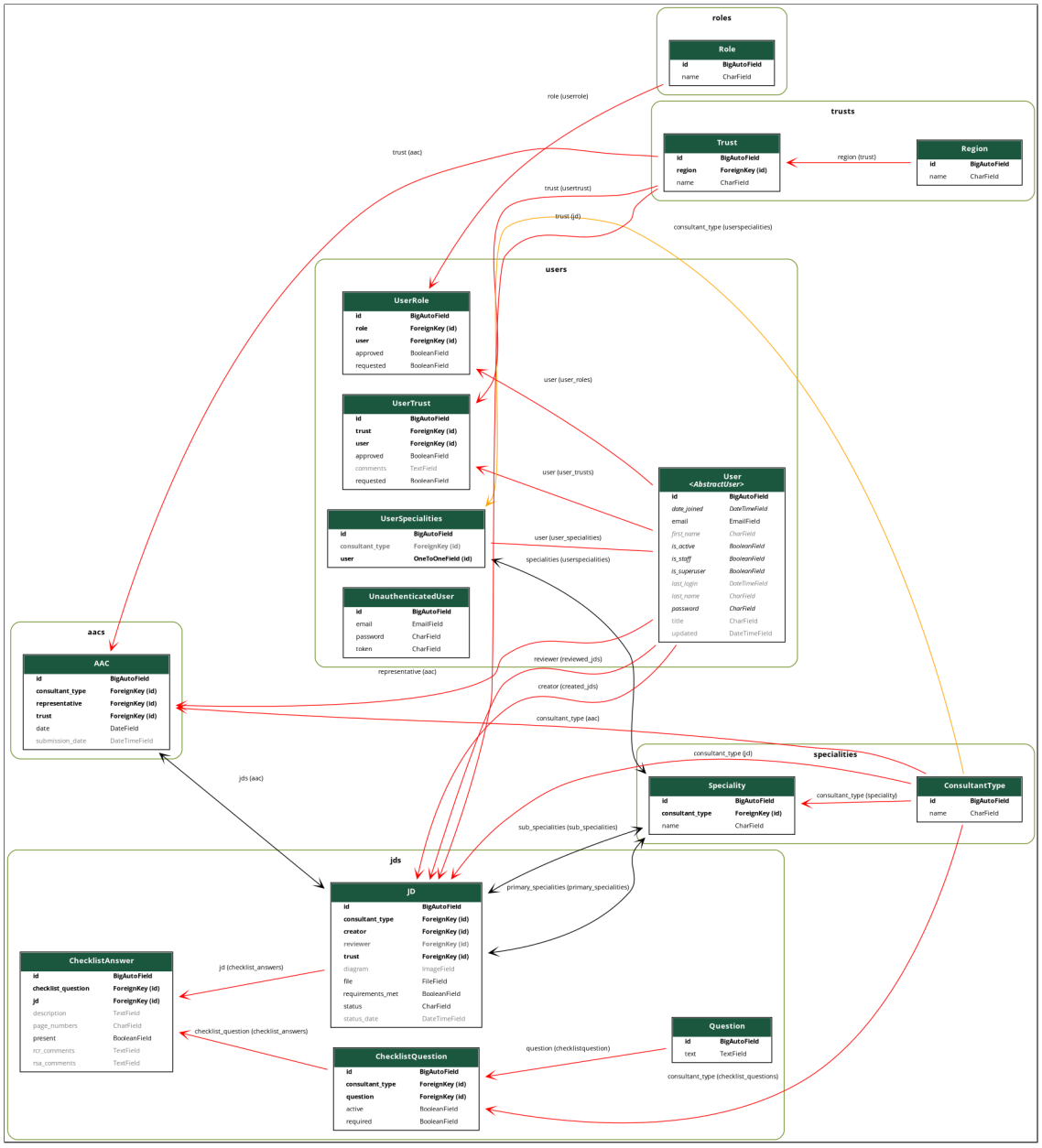
The Royal College of Radiologists (2022). *Clinical radiology job planning guidance for consultant and SAS doctors 2022*. URL: https://www.rcr.ac.uk/media/gzxjd252/rcr-publications_clinical-radiology-job-planning-guidance-for-consultant-and-sas-doctors_october-2022.pdf.

— (2023). *RCR statement on industrial action — The Royal College of Radiologists*. URL: https://www.rcr.ac.uk/news-policy/latest-updates/rcr-statement-on-industrial-action/#:~:text=There%20is%20currently%20a%2017 (visited on 04/24/2024).

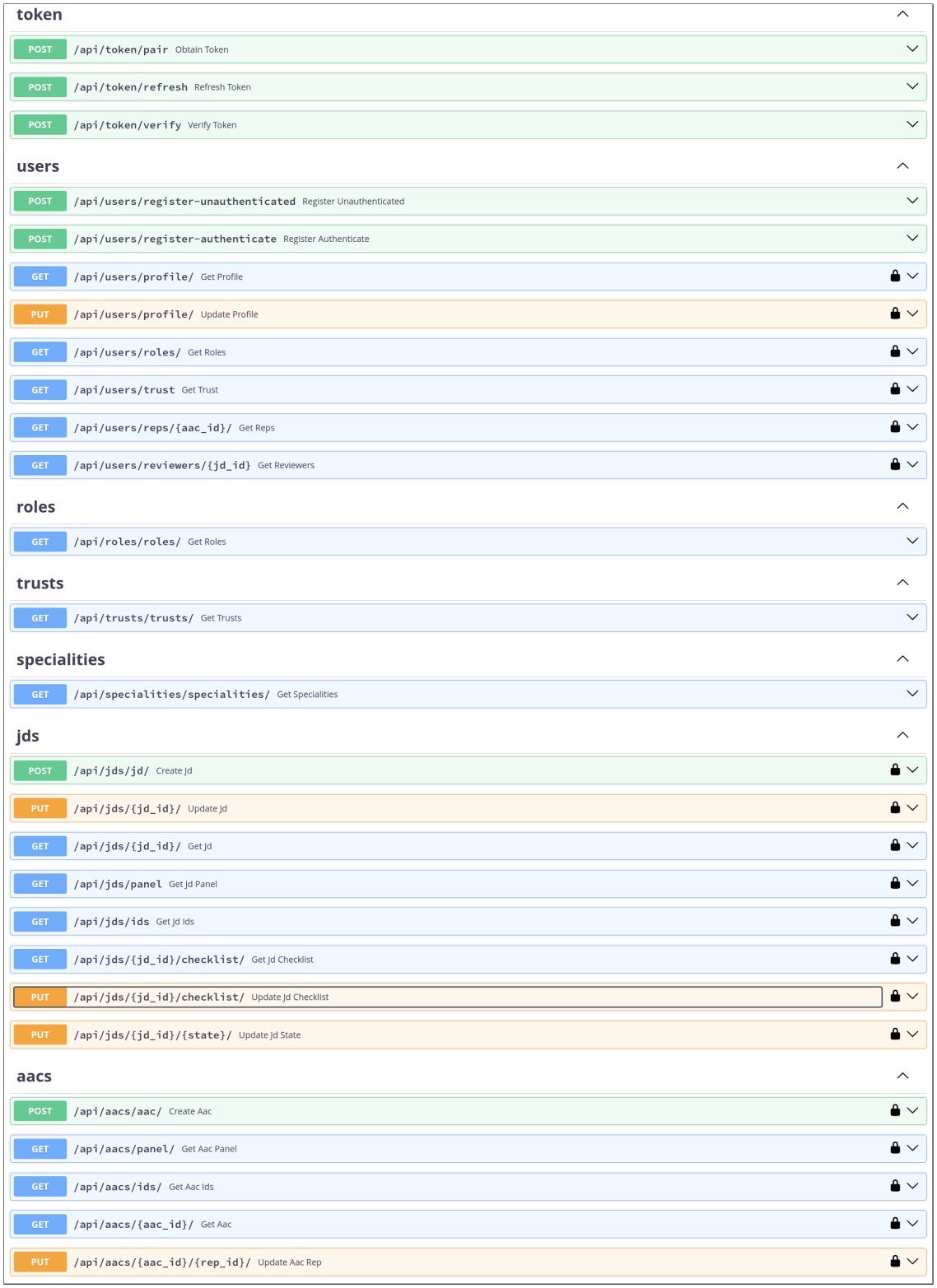
— (2024a). *Advisory Appointment Committee (AAC) — The Royal College of Radiologists*. URL: https://www.rcr.ac.uk/our-services/management-service-delivery/advisory-appointment-committee-aac/ (visited on 04/24/2024).

— (2024b). *Advisory Appointment Committee FAQs*. URL: https://www.rcr.ac.uk/our-services/management-service-delivery/advisory-appointment-committee-aac/advisory-appointment-committee-faqs/ (visited on 04/24/2024).

Appendix A: Database



Appendix B: API Endpoints



Appendix C: JD Checklist Endpoint