

Medical Transport Web Application with Optimization Routing

CE301: Individual Capstone Project
BSc Computer Science

Yi Kai, Liaw

yl23705@essex.ac.uk

Registration number: 2313797

Supervisor: Dr. Luca Citi

Second Accessor: Dr. Michael Barros

Abstract

Medical transport has been grappling with vehicle resource allocation and poor service quality for decades, yet a robust solution to address this issue remains elusive. This project presents a full-stack web application tailored for non-emergency medical transport, leveraging vehicle optimization routing to significantly enhance efficiency and reduce costs. The front end, built with React and Material UI, offers a modern and intuitive user interface for seamless interaction. On the backend, a robust Restful API powered by Django ensures the scalability and flexibility for future enhancements, with PostgreSQL serving as the database for efficient data management. Lastly, the core feature of vehicle optimization routing is facilitated through the integration of the Google Distance Matrix API and the Google OR-Tools Python package, ensuring optimized routing for medical transport vehicles. By combining these technologies, the application offers an unparalleled and comprehensive solution for managing medical transport logistics, streamlining operations, improving resource allocation, and ultimately improving the quality of service for both patients and healthcare providers to unprecedented levels of excellent

Table of Contents

1 INTRODUCTION	1
2 LITERATURE REVIEW	1
2.1 NON-EMERGENCY PATIENT TRANSPORT (NEPT).....	1
2.1.1 CURRENT APPROACH	2
2.1.2 CHALLENGES AND OPPORTUNITIES	3
2.2 GENETIC ALGORITHMS	3
2.2.1 ROUTE PLANNING THROUGH GENETIC ALGORITHM	3
2.3 VEHICLE ROUTING PROBLEM.....	4
2.3.1 VRP WITH MULTIPLE CONSTRAINTS	4
3 LEGAL, ETHICAL AND RELATED ISSUES	4
3.1 LEGAL ISSUES.....	5
3.2 ETHICAL CONSIDERATIONS	5
3.3 CULTURAL, SOCIETY AND RELATED MATTERS	5
3.4 SUSTAINABILITY.....	5
4 OBJECTIVES OF THE PROJECT.....	6
4.1 DEFINE THE SCOPES.....	6
4.2 DEFINE THE KEY FEATURES	7
5 IMPLEMENTATION OF THE WEB APP	7
5.1 USER INTERFACE DESIGN.....	7
5.2 OVERALL FOLDER STRUCTURE	8
5.3 FRONTEND	8
5.3.1 FOLDER STRUCTURE	9
5.3.2 HOME AND ABOUT US PAGE	9
5.3.3 OUR SERVICES PAGE.....	10
5.3.4 PATIENT ZONE.....	10
5.3.5 STAFF ZONE	13
5.4 BACKEND.....	16
5.4.1 MODEL.....	16
5.4.2 SERIALIZER	17
5.4.3 VIEWS	18
5.4.4 BROWSABLE API.....	18
5.4.5 AUTHENTICATION.....	19
5.4.6 PERMISSION.....	20
5.5 DATABASE.....	20
5.5.1 ER DIAGRAM	21
6 IMPLEMENTATION OF OPTIMIZATION ROUTING ALGORITHM.....	22
6.1 GOOGLE OR-TOOLS.....	22
6.2 ALGORITHM STRUCTURE	22
6.3 VULNERABILITIES	23
7 CONCLUSIONS AND FURTHER WORK	24
A PROJECT MANAGEMENT	24
A.1 PROJECT PROGRESS.....	25
A.2 WHAT HAS BEEN LEARNED	25
A.3 APPLICATION DEPLOYMENT	25

Chapter 1

1 Introduction

In today's healthcare landscape, efficient patient transportation is paramount for ensuring optimal care delivery. Non-Emergency Patient Transport Services (NEPTS) plays a crucial role in facilitating the safe and timely transfer of non-emergency patients to medical facilities and appointments. However, healthcare organizations often grapple with the complexities of managing NEPTS effectively, from coordinating schedules and resources to ensuring patient safety and satisfaction. To address these challenges and streamline the NEPTS management, this project introduces "Top Patient Transport", a comprehensive solution designed to revolutionize how healthcare provider plan, coordinate, and optimize patient transportation services.

The application is structured into three main sections: Public Zone, Patient Zone, and Staff Zone. In the Public Zone, users access NEPTS and other healthcare information. The Patient Zone features the service booking system for users, while the Staff Zone is dedicated to staff members' tasks and activities.

Additionally, this project's explanations are designed to be accessible to all, regardless of prior web application knowledge. Complex terminologies will be minimized, ensuring clarity for those new to the topic. However, having basic web application knowledge is beneficial for readers to grasp concepts more easily.

Last but not least, given the extensive scope of the project, attempting to cover every detail in this report would render it cumbersome to read. Therefore, the author will focus on elucidating the most significant components and essential information. If further exploration is desired, readers can refer to the codebase of the application and leverage the explanations provided herein as a guide.

Chapter 2

2 Literature Review

The review delves into Non-Emergency Patient Transport services (NEPTS), Genetic Algorithms (GA), and the Vehicle Routing Problem (VRP) within NEPTS route planning. It highlights current challenges and proposes solutions, such as GA optimization. Furthermore, it addresses the complexities of VRP with multiple constraints. This review paves the way for further exploration in subsequent chapters, stressing the importance of innovative solutions to improve the efficiency and reliability of NEPTS.

2.1 Non-Emergency Patient Transport (NEPT)

A non-emergency patient transport service is a service that provides free transport for eligible patients that requires to attend non-urgent treatments at an NHS site. Specifically intended for patients where other travel means might be detrimental to their conditions due to medical or mobility needs [7].

2.1.1 Current Approach

The current Non-Emergency Patient Transport Service (NEPTS) is provided by a range of public, private, and voluntary sector organizations, including NHS Ambulance Trusts and independent providers. The service comprises four main components:

- (i) **Coordination and triage capacity:** This involves evaluating eligibility, coordinating and overseeing journeys, and directing individuals to independent transport options.
- (ii) **Specialist transport services :** More than 300 registered ambulance providers deliver these services, offering adapted vehicles and trained staff support for patients with specific needs.
- (iii) **Non-specialist services:** Private hire/taxis and community support also play an important role in providing flexible transport solutions for individuals with less severe needs in certain areas.
- (iv) **Reimbursement:** The Healthcare Travel Costs Scheme assists patients and low-income individuals in covering the expenses of private transport.

Report indicates that NEPTS plays a crucial role in facilitating 11-12 million patient journeys annually in the UK, with an estimated expenditure of approximately £460 million per year from the government [3]. However, the total transport operation emits about 57-65 kilotonnes of carbon dioxide equivalent emissions annually, constituting roughly 20% of the NHS' direct travel emissions. Due to the diverse reasons for using the service, various vehicles and support are required to deliver NEPTS, such as:

- **High dependency ambulances** - at least two trained staffs.
- **Stretcher and specialist ambulances** – at least two trained to first aid level staffs.
- **Sitting and wheelchair accessible ambulances** – at least one trained staff.
- **Cars** – one trained ambulance care assistant or volunteers.
- **Minibuses** – one trained ambulance care assistant as driver.
- **Taxis and private hire vehicles** – one driver with some training.

Given that NEPTS is provided by various licensed organizations, different approaches are often implemented by each provider, leading to inconsistencies in service delivery. Consequently, patients frequently encounter difficulty accessing reliable information about the service, with a significant portion remaining unaware of NEPTS entirely. Typically, patients must contact the provider or visit their website, apply to determine eligibility, and then book the service accordingly. Figure 1 presents a simplified framework of the NEPTS:

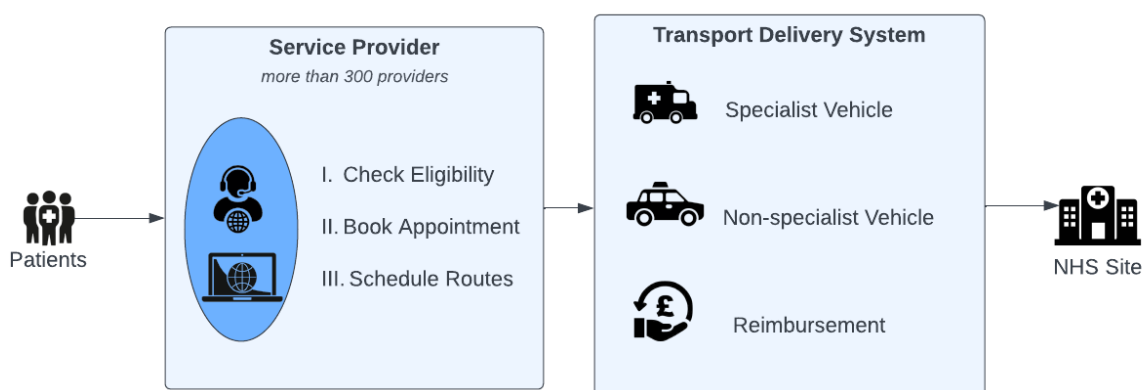


Figure 1. Simplified NEPTS Framework

2.1.2 Challenges and opportunities

The current system often results in nearly a third of patients waiting over three hours for transport back from treatment, leading to significant time of wastage. Patients frequently experience uncertainty regarding transport arrival times, resulting in needless waiting and anxiety. Moreover, some regions suffer from an alarming rate of around a quarter of journeys being cancelled or aborted annually, that is a total of nearly 3 million trips, highlighting poor coordination between providers and healthcare services. Additionally, data from five of the nine largest NEPTS providers reveals that approximately 5% of inbound patients arrive after their appointment time, as reported by the NHS [4]. In summary, the primary challenges encountered by most NEPTS users including:

- **Unreliable booking process**
Booking or modifying appointments via phone is challenging. Some getting frustrated with being asked the same questions again and again for eligibility check.
- **Inconsistency pickup delivery time**
Patients were often experiencing early or delayed pickups, some even encounters abrupt cancellations, which sometimes exacerbate their pain and anxiety.
- **Uncertain Information**
Lack of access to booking details and insufficient information about available transport vehicles are also common issues faced by users.

Various strategies have been proposed to enhance the reliability of NEPTS, including establishing a new national framework and leveraging technology for improved coordination. However, the efficacy of such initiatives remains uncertain, requiring time for both service providers and users to acclimate to the new frameworks. Moreover, without addressing user-end issues, the underlying problems may persist. Therefore, this report proposes the development of a NEPTS web application as a comprehensive solution to these challenges.

2.2 Genetic Algorithms

Genetic Algorithm (GA) is a subset of evolutionary algorithms based on Darwinian evolution theories, operate within a population where natural selection favours the fittest individuals, passing on advantageous traits to offspring. These individuals possess external properties influencing their fitness, or phenotype, along with a genotype comprising a set of genes encoding these traits.

GA employs the principal of natural selection to tackle complex optimization problems. In particular, each individual in a population represents a solution encoded within a chromosome associated with a fitness function, indicating its adaptability to the problem at hand. Through crossover and mutation operations, the fittest individuals' characteristics are then inherited by their successive generations, ensuring variability and adaption within the population. Classical GAs typically begins with a randomly generated initial population, progressing through generations via crossover and mutation to get the evolved and refined solution [5].

2.2.1 Route planning through genetic algorithm

The route planning for NEPTS involves scheduling daily routes for available ambulances, specifying stops, estimated arrival times, and patient pick-up and drop-off points. This task is typically executed by human experts who are aware of the limitations of the system and constraints that should be addressed to find feasible solutions. However, due to the sheer complexity of the problem, achieving optimal solutions manually is impractical. The distribution of travel journeys among available

ambulances poses a computationally intensive challenge. For instance, scheduling 50 journeys across 4 available ambulances would result in approximately 4^{50} possible combinations, which is 1.27×10^{30} possibilities. Even with a brute force algorithm capable of checking 10,000 combinations per second, it would take around 4×10^{18} years to evaluate all the possibilities. Obviously, such an approach is untenable, necessitating the exploration of alternative methods.

GA emerge as a promising solution to this dilemma. As mentioned previously, GA would efficiently navigate the vast search space of route combinations, focusing on solutions with positive outcomes., which enhances NEPTS route planning effectiveness and efficiency, leading to improved service delivery and patient outcomes [1].

2.3 Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is one of the most common optimisation problems. It encompasses a depot, a set of locations, and customer with known demands, as well as a fleet of vehicles. The primary aim is to devise the most cost-effective routes for multiple vehicles to traverse all locations, often considered with route length constraint. Through careful analysis and strategic planning, solutions to the VRP could significantly enhance operational efficiency and resource utilization, especially in the NEPTS [6].

2.3.1 VRP with multiple constraints

As discussed, the VRP encompasses not only route length constraints but also various other constraints that are crucial for route planning. In the context of NEPTS, the constraint to be considered include:

- **Pickups and deliveries:** This involves transporting items from the depot to designated locations and returning, ensuring all items are delivered correctly.
- **Time windows:** Planning schedules to visit locations that are only accessible during specific time frames, with the least route length.
- **Capacity constraint:** Each vehicle has a maximum capacity limit, necessitating careful consideration to minimize costs while adhering to capacity constraints during pickups and deliveries.

The VRP presents significant complexity within the context of NEPTS. Considering numerous constraints for each route, the algorithm often struggles to find solutions, highlighting the need for a meticulous design. Further details will be provided in subsequent chapters [2].

Chapter 3

3 Legal, Ethical and Related Issues

This chapter delves into the broader implications of the project's work within the global context. It explores its connection to pertinent aspects such as legal and ethical, while also addressing other relevant considerations. Furthermore, it evaluates the project's contribution to the overarching sustainability of the world, encompassing environmental, social, and economic dimensions.

3.1 Legal issues

Legal considerations play a pivotal role in the project's development, manifesting in three key areas. Firstly, the project relies on a suite of open-source development tools, including React, Django, and Material UI, all of which are governed by the permissive MIT License. This license facilitates unrestricted usage for both personal and commercial purposes, thereby mitigating any legal barriers associated with their incorporation.

Secondly, the images and icons utilized within the application are sourced from Google Images, meticulously filtered to adhere to the appropriate Creative Commons licenses. Given that the project is not intended for commercial use, this approach ensures compliance with legal requirements without encountering any obstacles.

Lastly, the integration of Google OR-tools to enable the route planning functionality is subject to the Apache 2.0 License. This license grants users the liberty to freely utilize, modify, distribute, and sublicense the software, even for commercial endeavors, thereby eliminating any potential legal constraints in its implementation.

3.2 Ethical considerations

In developing the web application, several ethical considerations are prioritized to uphold patient privacy, data security, and equitable access to healthcare services. Central to its approach is obtaining explicit consent from patients for the collection, use, and storage of their personal information, fostering transparency and informed decision-making. Additionally, the application upholds transparency in data usage, accuracy of information, user empowerment, and ethical conduct in handling patient data throughout the development and deployment process. Moreover, it prioritizes data security by implementing encryption and secure authentication mechanisms.

Equitable access to NEPTS is also facilitated by making the application accessible to all users, offering multiple booking channels, and addressing the needs of marginalized communities. Through these ethical principles, the project aims to foster trust, accountability, and responsible use of patient data, ultimately enhancing the efficiency and reliability of NEPTS while prioritizing patient well-being and confidentiality.

3.3 Cultural, society and related matters

Since this project is designed to enhance the reliability of NEPTS in the UK, eligibility checks before granting access to certain features within the application is necessary. These checks are primarily based on the user's health condition, with factors such as gender and nationality treated solely as personal details of the user. In addressing concerns regarding racial bias, the application ensures equal access to NEPTS regardless of race. However, due to the fact that it is specifically designed for use cases in the UK, it means that individuals from other countries may not have access to all features of the application.

3.4 Sustainability

Sustainability serves as a fundamental pillar in the development of the NEPTS web application. In addressing sustainability within this context, several key considerations emerge:

- (1) **Environmental Impact:** The NEPTS web application would reduce unnecessary emissions and fuel consumption. By facilitating efficient scheduling and route planning, the application contributes to mitigating the carbon footprint associated with patient transport services. Also,

the utilization of digital platforms minimizes the reliance on paper-based processes, further diminishing environmental impact.

- (2) **Social Responsibility:** Beyond environmental concerns, the application prioritizes social responsibility by enhancing accessibility in healthcare services. By streamlining the booking and transportation process for NEPTS, it ensures that individuals with diverse mobility needs can access essential healthcare services conveniently.
- (3) **Economic Efficiency:** Through effective route planning and utilization of available transportation resources, the application maximizes efficiency, reducing unnecessary expenditure and enhancing cost-effectiveness in healthcare service delivery. Moreover, the digitization of processes minimizes administrative overheads and enhances overall operational efficiency, contributing to long-term financial sustainability.
- (4) **Long-term Viability:** By incorporating modular design principles and scalable infrastructure, the application can adapt to evolving technological landscapes and accommodate future growth seamlessly. Furthermore, ongoing monitoring and evaluation mechanisms enable continuous improvement, ensuring that the application remains aligned with sustainability objectives and addresses emerging challenges effectively.

Chapter 4

4 Objectives of the project

The project aims to create a robust and comprehensive web application capable of addressing the challenges outlined in Chapter 2, with a primary focus on implementing booking management functionality and integrating an optimized routing algorithm. Given the complexity of the topic, expanding the project scope extensively could render it impractical to complete within a very limited timeframe. Therefore, it is essential to narrow down the project scope and evaluate its feasibility for timely completion.

4.1 Define the scopes

While the NEPTS indeed comprises massive and intricate data, the imperative to narrow down the scope primarily stems from the complexity of the schedule routing algorithm. There wasn't sufficient time to develop a routing algorithm capable of accommodating every single potential event within the NEPTS routing scenarios, this will be further discussed in Chapter 5.

The routing algorithm is designed under certain rules:

- Only receive location data in Colchester (only provides transport service within Colchester)
- Only one type of vehicle available, special vehicles are not considered (all vehicle has the same capacity)
- Transport service time is between 8.00 AM – 4.00 PM (BST) everyday
- At most one (1) escort for each patient

4.2 Define the key features

Patient Zone

- Access latest NEPTS and healthcare information
- Register account and Login to the NEPTS portal
- Manage account details (update personal details)
- Manage NEPTS bookings (create, view, update, cancel bookings)

Staff Zone

- Login as staff using staff id
- Approve/reject patient's NEPTS eligibility
- Manually add NEPTS booking for elderlylies
- Generate and view routing schedule

Chapter 5

5 Implementation of the Web App

This chapter delve into the development details of the web application, highlighting the tools and technologies utilized. The synergy between design and development is also emphasized, showcasing how creativity and technical proficiency merge for an intuitive user experience.

5.1 User interface design

The user interface (UI) is designed using **Figma**, this includes every single page of the web app. The design process involved the consideration of themes, page-to-page navigation, and overall aesthetics. Given that a significant portion of the user base consists of older individuals, simplicity and straight forward were the primary focuses throughout the design process, aiming to ensure ease of use for all users.

Though it took some time to come up with the design, this comprehensive approach not only streamlines development but also provides a clear visualization of the final product. The coding largely aligns with the UI design, ensuring consistency and efficiency throughout the development process. Figure 2-4 below provide a glimpse of the UI design in Figma.

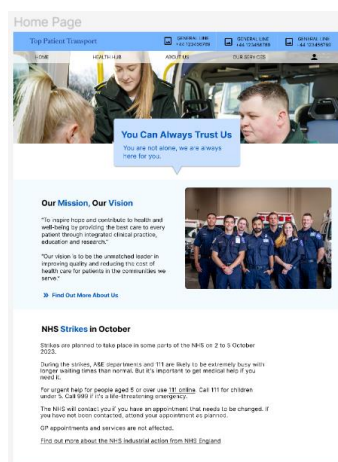


Figure 2. UI Design 1

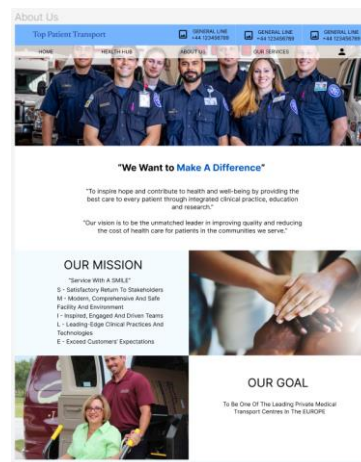


Figure 3. UI Design 2

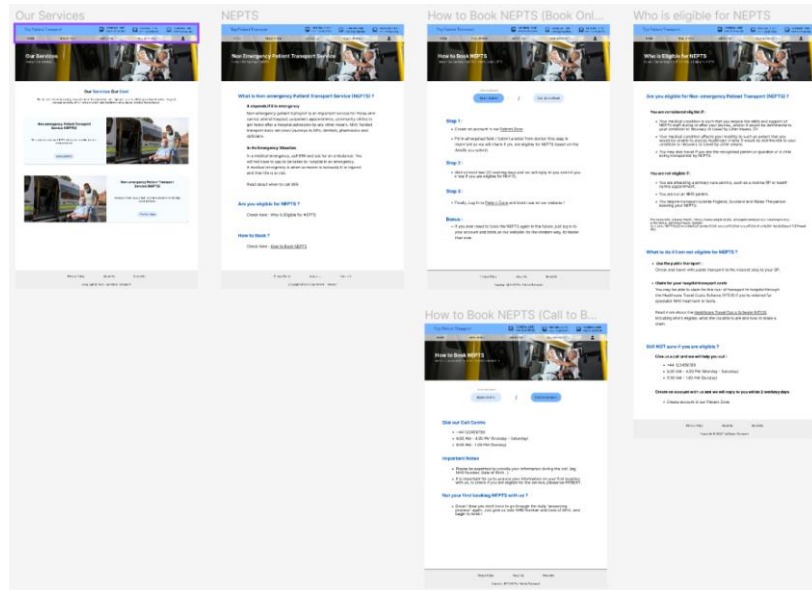


Figure 4. UI Design 3

5.2 Overall Folder Structure

The overall folder is structured aligning with **Django web framework**, separated in two main components, backend and frontend, where the folder name for backend is named as “api” and “frontend” for frontend (as illustrated in figure 5). All frontend related files and components are stored within the “frontend” folder, while backend related files are stores within the “api” folder, as shown in figure below. The details will be discussed step by step later in this chapter.

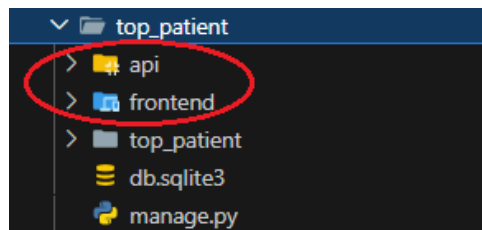


Figure 5. Overall Folder Structure

5.3 Frontend

The frontend is developed entirely using React Class Components. **React** is one most the most famous JavaScript libraries for building user interfaces, particular for single-page applications (SPA) with complex UI. One of the key features of it is the component-based architecture which enable developers to build complex UIs by breaking them into smaller, reusable pieces. The utilization of a virtual DOM (Document Object Model) also improves performance by minimizing the number of updates to the actual DOM of browser, resulting in faster rendering speeds.

Additionally, **Material UI** is used to even enhance the efficiency of development. It is a react component library designed by google that implements the Google’s Material Design guidelines. The material design emphasizes a clean, modern look with a focus on usability and user experience consistency across different platform and devices. Most importantly, it allows developers to quickly build attractive and functional UIs in React.

Due to time constraints, responsiveness is only partially implemented, meaning the web application may not perfectly adapt to every screen size.

5.3.1 Folder Structure

Due to the SPA nature of React, the entire application is contained within a single HTML page, dynamically rewriting the page rather than loading the entire new pages from the server. As shown in figure 6, all pages and reusable components in this application are stored within the “src” folder, and will be rendered in the “index.html” page.

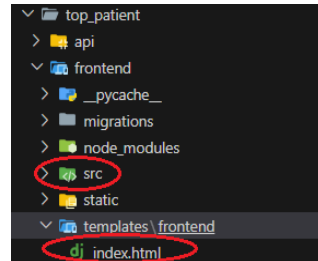


Figure 6. SPA Structure

In the “src” folder, reusable components and web pages are organized into specific folders like "components", "Main", "OurServices", etc. Notably, the "Patient Zone" and "Staff Zone" folders encompass all pages dedicated to managing NEPTS bookings, the details will also be explained in this chapter. Figure 7 illustrates the folder structure of the frontend folder.

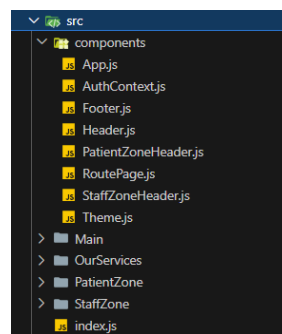


Figure 7. Frontend Folder Structure

5.3.2 Home and About Us page

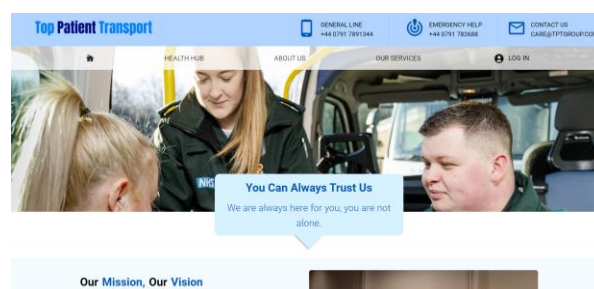


Figure 8. Home Page

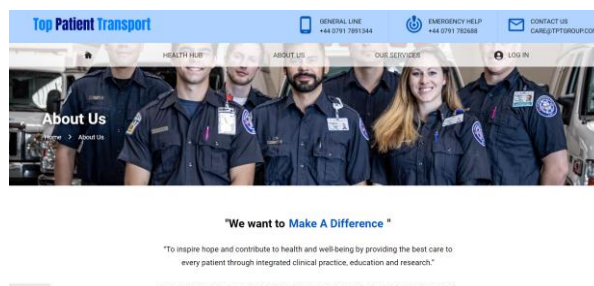


Figure 9. About Us Page

Home page (figure 8) serves as the landing page for users accessing the web application. It provides a concise overview of the services offered, latest updates, frequently asked questions, and more, tailored to inform users about the information and services available within the application. Subsequently, the About Us page (figure 9) offers a straightforward explanation of the organization's mission, goals, and activities.

Apart from that, it is worth noting that some reusable components like page header and footer are imported from the “components” directory, this eliminates the need to rewrite these components for each individual page, streamlining development and ensuring consistency throughout the application.

5.3.3 Our Services page

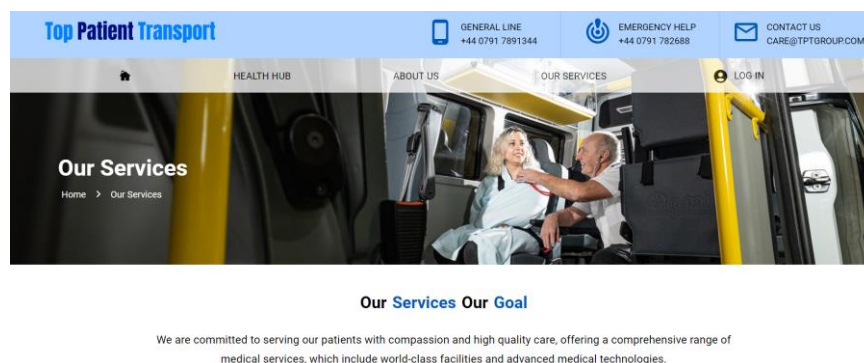


Figure 10. Our Service Page

As discussed in Chapter 2, patients frequently encounter difficulties in obtaining accurate information about NEPTS. The Our Services pages (figure 10) are meticulously designed to address this issue directly. Their purpose is to serve as a comprehensive information hub, offering users access to all pertinent NEPTS details, such as eligibility criteria, available services, booking procedures, and more. Furthermore, these pages play a crucial role in educating users about the service, thereby empowering them to make informed decisions and utilize the service effectively. The relevant pages reside within the “OurServices” folder, as in figure 11.

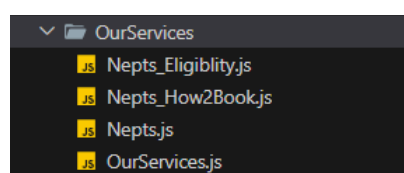


Figure 11. Our Services files

5.3.4 Patient Zone

The Patient Zone addresses another issue highlighted in Chapter 2, where patients often complain about the unreliability of the NEPTS booking system. This section provides users with access to all available services through the web application, of course, only NEPTS is available for now.

5.3.4.1 Register and Login

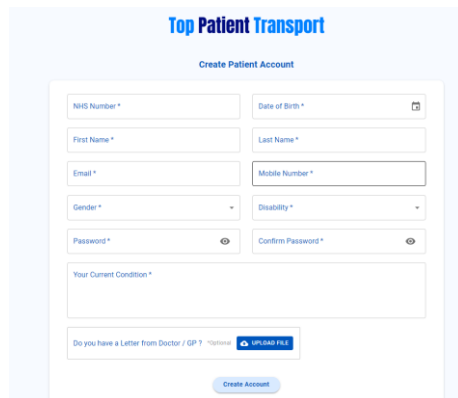
The 'Create Patient Account' form for 'Top Patient Transport' includes fields for NHS Number, Date of Birth, First Name, Last Name, Email, Mobile Number, Gender, Disability, Password, and Confirm Password. It also has a text area for 'Your Current Condition' and a checkbox for 'Do you have a Letter from Doctor / GP?'. A 'Create Account' button is at the bottom.

Figure 12. Patient Register

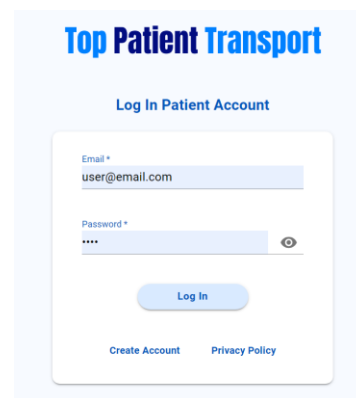
The 'Log In Patient Account' form for 'Top Patient Transport' includes fields for Email and Password. It features a 'Log In' button and links for 'Create Account' and 'Privacy Policy' at the bottom.

Figure 13. Patient Login

Before using the services, the users are required to register and login to (figure 12 & 13) the website. The patient details will then be validated by the staffs, we will see that very soon in this Chapter.

5.3.4.2 Dashboard

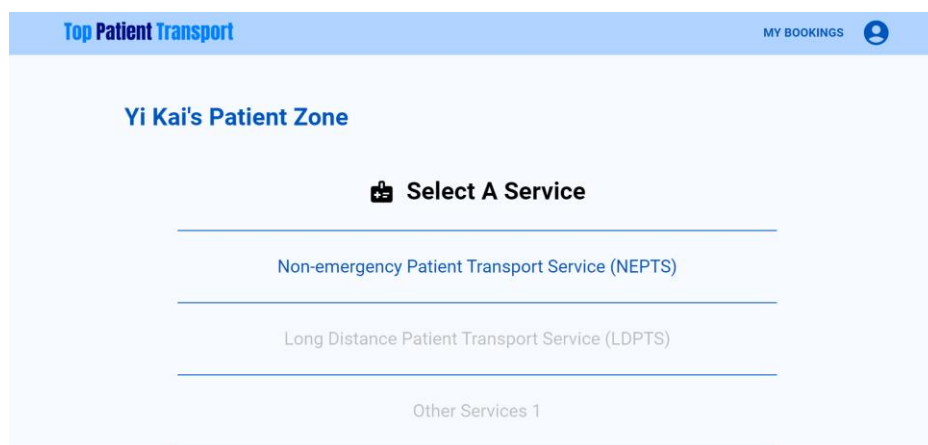
The 'Yi Kai's Patient Zone' dashboard for 'Top Patient Transport' features a 'MY BOOKINGS' link and a 'Select A Service' section. The services listed are 'Non-emergency Patient Transport Service (NEPTS)', 'Long Distance Patient Transport Service (LDPTS)', and 'Other Services 1'.

Figure 14. Patient Dashboard

After logging in, user will be redirected to the patient dashboard page (figure 14). As mentioned earlier in this chapter, the UI design prioritizes simplicity and practicality. Therefore, aside from the core portal functionality, no additional features have been included.

From the dashboard, users can select available services to book appointments. Currently, only NEPTS is available, the dashboard design also ensures the scalability for future expansions of the platform.

5.3.4.3 Profile

Top Patient Transport MY BOOKINGS

Yi Kai's Profile

NHS Number : 1234567890

Gender : Male

Birth Date : 2001-03-15

Disability : I'm able to Walk

Contact : 0194143449

Email : user@email.com

Update Changes

Figure15. Patient Profile

The patient profile page (figure 15) displays the user's personal details, including their NHS Number, Gender, and Disability, which will automatically serve as the default value during NEPTS booking. Editing is disabled for NHS Number and other static details, while contact information can be edited if necessary, considering the likelihood of phone number changes.

5.3.4.4 NEPTS Booking

Top Patient Transport MY BOOKINGS

Non-Emergency Patient Transport Booking (NEPTS)

Approval Pending

Your account is still pending for our staff to validate your eligibility to use our NEPT Service.

We will inform you via email once your account has been validated. Contact our general line +44 123456789 if you have further enquiries.

Home Page

Figure 16. NEPTS Pending Approval

Top Patient Transport MY BOOKINGS

Non-Emergency Patient Transport Booking (NEPTS)

Not Eligible

Sorry, you are not eligible for using our NEPT Service.

Contact our general line +44 123456789 if you need further assistance.

Home Page

Figure 17. NEPTS Rejected

As mentioned earlier in this chapter, user eligibility for services will be validated by staff after account creation. While waiting for approval, users will not be able to access services. The NEPTS booking page will display the validation status (pending or rejected) along with contact details if they ever wish to appeal. As shown in Figure 16 and Figure 17.

Top Patient Transport MY BOOKINGS

Non-Emergency Patient Transport Booking (NEPTS)

Date & Time Location Appointment Details Done

Appointment Date * 04/25/2024

Appointment Time * 02:00 PM

Appointment End Time (Est.) * 03:00 PM

Next Step

Figure 18. NEPTS Booking Form

Top Patient Transport MY BOOKINGS

Non-Emergency Patient Transport Booking

Booking request submitted!

Date & Time Location Appointment Details Done

Booking Reference Number

JUXLPXKB

Thank you for booking with Top Patient Transport. Your request has been successfully submitted and we've sent an confirmation email to you. We look forward to seeing you, have a pleasant day ahead!

Kindly note that cancelling your appointment last minute would delay other patient's schedule, we appreciate your understanding. Contact our general line +44 123456789 if you have further enquiries.

Home Page

Figure 19. NEPTS Submitted

After approval, users now gain access to the booking form (figure 18). They will provide the appointment details in a step-by-step basis. Required information includes appointment date and time, estimated end time, pick-up address, medical centre address, and escort need. Error checking is implemented to ensure all fields are filled before proceeding. Note: Only times after the “appointment time” can be selected for the “appointment end time” field.

Once submitted, a unique 8-alphabet booking reference number is generated accompanied by important notes, as illustrated in figure 19. The 8-alphabet reference number offers ample combinations, with 26^8 possibilities, satisfying the application's needs and ensuring uniqueness.

5.3.4.5 Manage Appointments

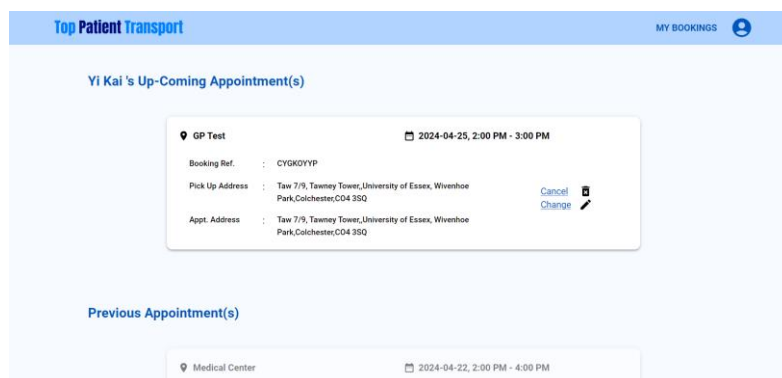


Figure 20. Managing Appointments

After submitting the booking request, users can now manage appointments on the ‘My Bookings’ page (figure 20). Users will be able to view the upcoming appointments and past expired appointments. The appointments are sorted based on the nearest appointment dates.

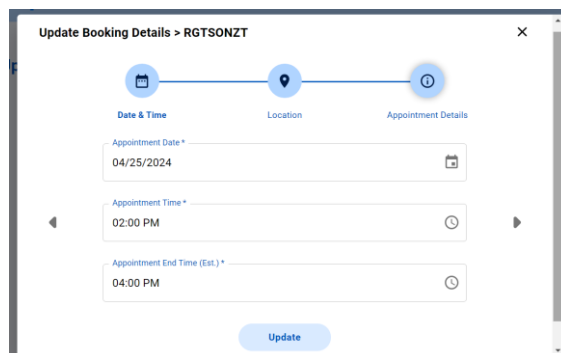


Figure 21. Update Appointment Detail

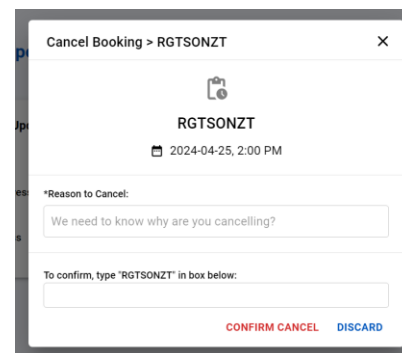


Figure 22. Cancel Appointment

Additionally, users have the option to update (figure 21) or cancel (figure 22) their appointments. To prevent accidental cancellations, an extra confirmation step is required for appointment cancellations, where users must provide a reason for cancellation and type the booking reference number.

5.3.5 Staff Zone

The Staff Zone is implemented to enhance the reliability and completeness of the system. Here, staff members can manually add NEPTS bookings for elderly individuals, validate user eligibility for services, and most importantly, access to the optimized vehicle routing schedules.

5.3.5.1 Dashboard

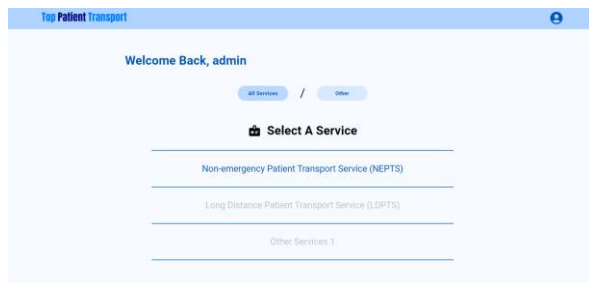


Figure 23. Staff Dashboard – All Services

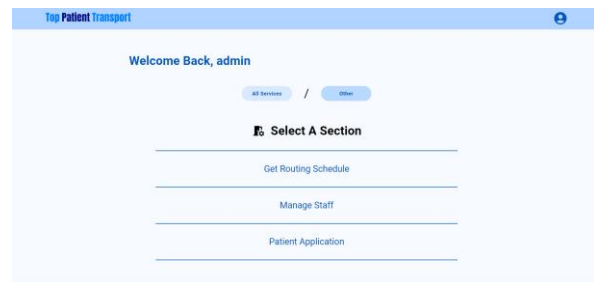


Figure 24. Staff Dashboard - Other

The staff dashboard (figure 23 & 24) differs slightly from the patient dashboard. In addition to manually adding service bookings for elderly users, staff members have access to features for obtaining optimized routing schedules and validating user eligibility. However, the ‘manage staff’ feature is not yet fully implemented, it has been designed for future scalability.

5.3.5.2 NEPTS Booking

Figure 25. NEPTS Booking Form

Similarly to the patient NEPTS booking form, staffs will have to fill in the appointment details step-by-step. There is only one additional step for staff: manually adding the patients’ personal details, since these call-in patients’ details are not recorded in the database.

5.3.5.3 Patient Application

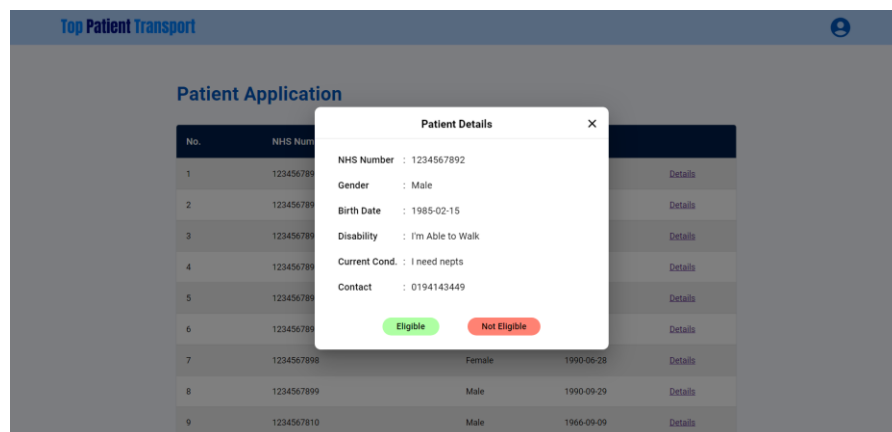


Figure 26. Validate Users’ Eligibility for NEPTS

This feature prevents service abuse and prioritizes limited resources for those truly in need. The ‘patient application’ (figure 26) page displays a list of newly registered users, allowing trained staff to review patient details and health conditions to validate the user’s eligibility for service use. The result will be reflected on the patient NEPTS booking form as mentioned slightly earlier.

5.3.5.4 Get Routing Schedule

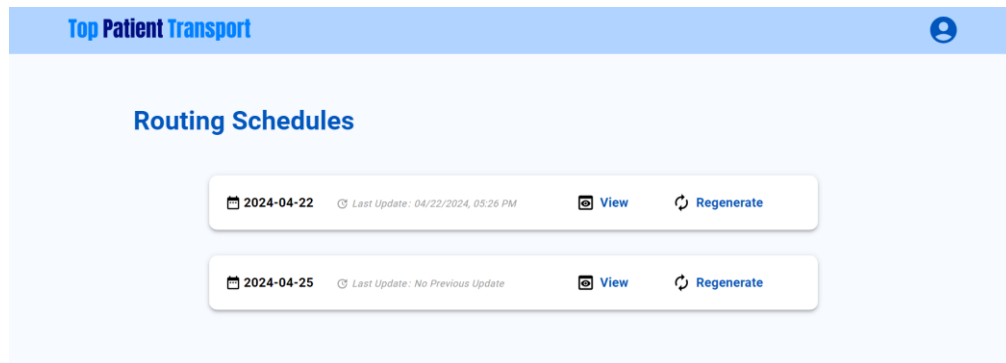


Figure 27. Routing Schedule

This is where staff access vehicle routing schedules. Only schedules for the next five days are displayed, as depicted in Figure 27. Notably, if there are no appointments on a specific date, it will not be listed here.

The ‘Last Update’ attribute indicates when the schedule for that day was generated or updated. Routing schedules are not automatically generated; staff members must press the ‘View’ button to generate the schedule for a specific day. For instance, on the date ‘2024-05-25’, it shows ‘No Previous Update’ as the schedule has not yet been generated for that day. On the contrary, if the schedule for a date, say ‘2024-04-22’, has already been generated previously, users can also press ‘Regenerate’ to update the schedule if new appointments are added.

The process of generating the schedule will be discussed in detail later in Chapter 6. The figure below provides details of the generated vehicle routing schedule. To better view the details for the schedule below, please follow this link for the [PDF file](#).

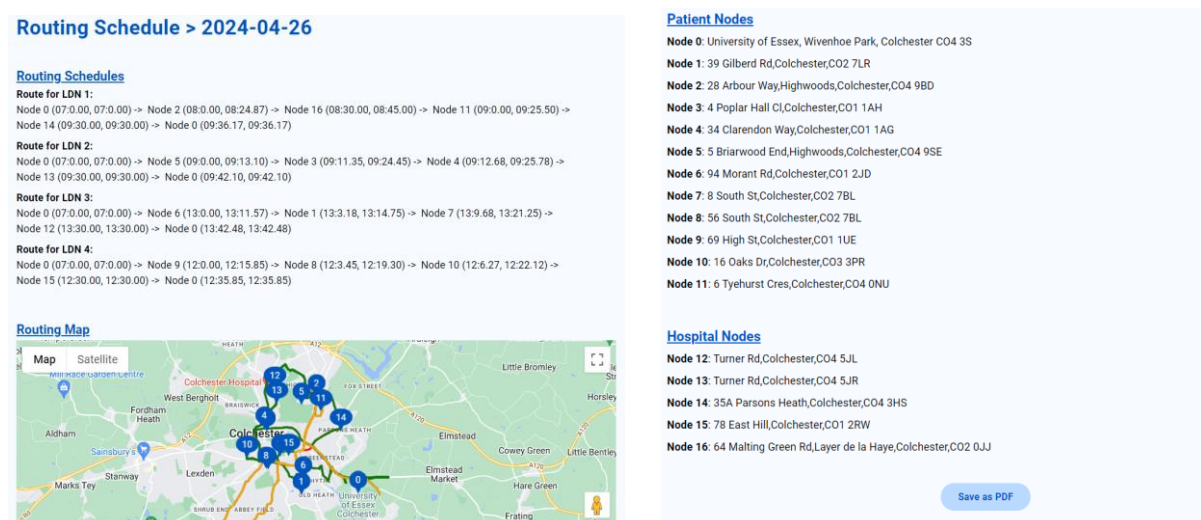


Figure 28. NEPTS Vehicle Routing Schedule

The schedule comprises of four main components, as explained below:

(1) **Routing Schedules:**

Shows the optimized route planning for each vehicle in an ordered node list. Node 0 represents the starting location, can be referred to as the organization's headquarters, selected as the University of Essex in this case.

Taking the route for vehicle 'LDN 1' as an example, it begins at Node 0 at 7 AM. The first stop is Node 2, where it should arrive between 8 AM and 8:25 AM to avoid delaying subsequent schedules. This pattern continues until the vehicle returns to the headquarters after completing all stops.

(2) **Routing Map:**

Google Map that visually represents each vehicle's route using distinct colours.

(3) **Patient & Hospital Nodes:** Each node represents a location, either for a patient's pick-up address or a medical centre.

5.4 Backend

The backend utilizes the **Django REST framework (DRF)**, a powerful toolkit for constructing Web APIs in Python with the Django web framework. It significantly streamlines the creation of RESTful APIs, facilitating rapid development and efficiency. In essence, DRF simplifies the process of building robust and scalable web APIs through its comprehensive features, elaborated further in this section. Figure 8 shows the folder structure in backend.

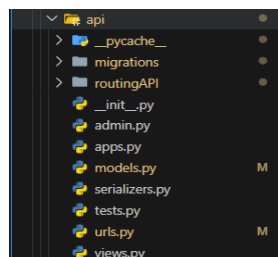


Figure 29. Backend folder structure

5.4.1 Model

The models serve as a fundamental component in defining the structure and behaviour of the application's data, encapsulating data attributes, relationships, validation logic, and database interaction. They are represented as Python classes, correspond to tables in a relational database management system (RDBMS) like PostgreSQL, MySQL, or SQLite, acting as a bridge between the database and Python code.

They allow developers to define entities and relationships specifying attributes like text, numbers, dates, and relationships with other models. Models also include validation logic to enforce data integrity, ensuring that data stored in the database meets specific criteria.

```
# Patient Model
class Patient(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    nhsNumber = models.CharField(max_length=10, validators=[MinLengthValidator(10)], default='0000000000', unique=True)
    gender = models.CharField(null=False, max_length=10, default='gender')
    birthDate = models.DateField(null=False, default=(2001, 3, 15))
    disability = models.IntegerField(null=False, default=1)
    contact = models.CharField(max_length=15, default="999")
    neptsValid = models.IntegerField(default=0, null=True)
    currentCondition = models.TextField(default='', null=False)
```

Figure 30. Patient Model Python Class

Taking figure 30 as example, it illustrates the definition of a Patient class, that directly corresponds to the Patient table within the database (database further elaborated later in this chapter). This table serves as a repository for storing patient information.

Several fields are explained as below. (other classes or Model adhere to the similar convention)

- (1) **`user`**: Represents a one-to-one relationship with Django's built-in `User` model, indicating that each patient is associated with a single user account. The 'on_delete' constraint ensures data integrity by removing associated patient data when the user account is deleted.
- (2) **`nhsNumber`**: A character field representing the NHS number of patient, constrained to a maximum of 10 characters. The unique constraint ensures each NHS number is unique in the database.
- (3) **`gender`**: A character field storing the gender of the patient with a default value of 'gender', and must not be empty.

5.4.2 Serializer

This is a crucial tool in DRF for converting complex data types, such as model instances or query sets, into native Python data that can be easily rendered into JSON, XML or other content types. It also handles the reverse process, converting parsed data back into complex types. Serializers determine the structure and format of data transmitted over APIs, allowing developers to specify which fields to include, how they should be represented, and how data should be validated upon deserialization. With serializers, developers can ensure data integrity, handle relationships between data sources, and customize data formats to suit the needs of their applications.

```
# Patient Serializer
class PatientSerializer(serializers.ModelSerializer):
    class Meta:
        model = Patient
        fields = '__all__'

# Add Patient Serializer
class AddPatientSerializer(serializers.ModelSerializer):
    class Meta:
        model = CombineUserPatient
        fields = ['email', 'password', 'first_name', 'last_name',
                 'nhsNumber', 'gender', 'birthDate', 'disability', 'contact', 'currentCondition']
```

Figure 31. Serializer Python Classes

Figure 31 illustrates two Python classes that defines the data structures transmitted over APIs in various scenarios. For patient registration, 'AddPatientSerializer' is defined to interact with the 'CombineUserPatient' model, specifying fields like 'email', 'password', 'first_name', and 'last_name'. During serialization, data is formatted for transmission and converted back into Python objects during deserialization.

For extracting patient data from the database, 'PatientSerializer' interacts with the 'Patient' model, specifying all fields for transmission.

Numerous serializers are implemented to accommodate various scenarios within the application, fostering smooth data interaction between the web application and the database, thereby ensuring efficient data transmission and manipulation.

5.4.3 Views

Views are Python functions or classes that receive incoming HTTP requests and generate corresponding HTTP responses. They encapsulate the application's business logic, interacting with data sources and rendering content for web pages or APIs. Views are mapped to specific URLs, directing requests to the appropriate logic layer within the application.

```
# Get Specific Patient Info
class GetPatient (APIView):
    def get(self, request, format=None):
        user = request.user
        patient = user.patient
        serializer = PatientSerializer(patient)
        return Response(serializer.data)
```

Figure 32. Patient Views Python Class

Various classes are defined to manage different scenarios within the application while adhering to similar conventions, thus it is important to understand the base structure. Figure 32 illustrates the retrieval of patient data from the database in response to GET requests. Upon receiving a GET request, the 'get()' method is triggered.

- (1) **First**, the method retrieves the user associated with the request using 'request.user'. If you remember, slightly earlier in this chapter we mentioned, each patient data is linked with a user through a one-to-one relationship, thus this step is crucial.
- (2) **Subsequently**, an instance of the 'Patient Serializer' is created, specifying the data structure for transmission and incorporating the patient data as an argument.
- (3) **Finally**, the method returns an HTTP response containing the serialized patient data in JSON format via the 'Response' class.

5.4.4Browsable API

One of the most notable features provided by DRF, which provides a web-based interface for interacting with the API directly from the web browser. It simplifies development and debugging by automatically generating HTML representations of API endpoints, enabling efficient building and testing of RESTful APIs within Django projects.

```
# Patients
path('create-patient', CreatePatient.as_view()),
path('get-patient', GetPatient.as_view()),
path('update-patient', UpdatePatient.as_view()),
path('get-patient-neptsReq', GetPatientNeptsReq.as_view()),
path('update-patientNepts/<userID>', UpdatePatientNEPTS.as_view()),
```

Figure 33. Browsable API – URL Patterns

Figure 33 illustrates the URL patterns for various views that handle different functionalities related to patients. Each path corresponds to a specific action or API endpoint for managing patient data. For example:

- (1) **`/create-patient`**: Maps to the 'CreatePatient' view, which is responsible for creating a new patient record.
- (2) **`/get-patient`**: Maps to the 'GetPatient' view, used for retrieving patient information.
- (3) **`/update-patient`**: Maps to the 'UpdatePatient' view, responsible for updating existing patient records.
- (4) **`/get-patient-neptsReq`**: Maps to the 'GetPatientNeptsReq' view, which retrieves information related to NEPTS requests for patients.
- (5) **`/update-patientNepts/<userID>`**: Maps to the 'UpdatePatientNEPTS' view, used for updating NEPTS information for a specific patient identified by <userID>.

5.4.5 Authentication

Authentication verifies the identity of users accessing the application. Django offers various authentication methods, including session-based authentication, token authentication, OAuth authentication, and others. Token authentication is implemented using JWT (JSON Web Token) in this application, where it involves issuing tokens to users upon successful authentication.

Tokens are typically long, randomly generated strings that serve as credentials for accessing protected resources. When a user logs in or authenticates, the server generates a token and sends it back to the client. The client includes this token in subsequent requests to authenticate itself. Tokens are stateless and can be stored on the client side (e.g., in local storage or cookies). In this project, it is stored in the local storage.

Storage	Key	Value
Local storage	django.admin.navSidebars...	true
http://127.0.0.1:8000	authToken	{"access": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0b2t1b190eXB1IjoieWVhbnZlZmVjZXNziwizXhwijoxNzEzOTkxNTU0LjYX...",
Session storage		
IndexedDB		
Web SQL		
Cookies		
Private state tokens		
Interest groups		
Shared storage		
Cache storage		

Figure 34. Access & Refresh Tokens

As illustrated in figure 34, two tokens are stored in the browser's local storage, detailed as follows:

- (1) **Access Token:** A short-lived credential issued to the client. The ‘actual token’ that grants use permission to access information from the database, such as profile and appointment details. Its short lifespan, typically set to minutes or hours (e.g., **15 minutes** in this application), ensures security, just like the bank apps. When the token expires or is invalidated, the client must re-authenticate with the server, to gain new access token.
- (2) **Refresh Token:** A long-lived credential issued to the client alongside with the access token upon successful authentication. Unlike the access token, which has a short lifespan, the refresh token persists longer (e.g., **15 days** in this application) and serves to obtain a new access token before the current one expires or becomes invalid. In this application, it is set to automatically requests a new access token from the authentication server every **8 minutes**. This mechanism maintains user sessions and ensures a seamless experience, minimizing the need for frequent re-authentication and at the same time, preventing unauthorized access.

```
[24/Apr/2024 22:04:42] "POST /api/token/ HTTP/1.1" 200 641
[24/Apr/2024 22:12:39] "POST /api/token/refresh/ HTTP/1.1" 200 641
[24/Apr/2024 22:20:39] "POST /api/token/refresh/ HTTP/1.1" 200 641
```

Figure 35. Server Log for refreshing tokens

Figure 35 illustrates the server log for API token updates. As seen in the first row, upon login, the client will continuously request new access token using the refresh token every **8 minutes**, as long as the user did not end the session (logout). Additionally, if the user does not log in to their account for **15 days**, the refresh token expires and the user will have to re-authenticate with the server.

5.4.6 Permission

Permissions determine what actions authenticated users are allowed to perform within the application. Django provides a flexible permissions system, allowing developers to define custom permissions or use built-in ones such as 'IsAuthenticated' (grants access to authenticated users) and 'IsAdminUser' (grants access to superusers). However, nothing complex is implemented for permissions, the only rule is, users must be authenticated in order to gain access to the database information.

As mentioned earlier in the authentication section, access token serves as the 'access card' to retrieve information from the database.

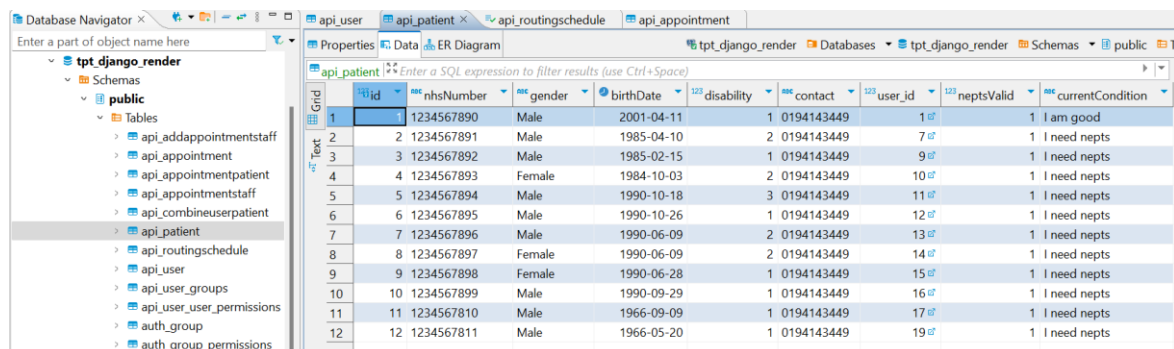
5.5 Database

PostgreSQL is chosen as the database management system (DBS) for this project, it is often considered the best choice to use with Django projects mainly for two reasons.

Firstly, PostgreSQL's compatibility with Django is seamless, as Django provides built-in support for PostgreSQL databases, making it easy to integrate and work with within Django projects. This native support ensures efficient data management and interaction between Django's ORM (Object-Relational Mapping) layer and the PostgreSQL database, significantly simplifying development and tasks maintenance.

Secondly, as a relational database management system (RDBMS), it is highly scalable and capable of handling large volumes of data and concurrent connections. This scalability is essential in consideration for the future growing of the application as it will need to accommodate increasing user traffic and data demands over time.

Figure 36 below illustrates a portion of data stored in the database for the 'Patient' table.



id	nhsNumber	gender	birthDate	disability	contact	user_id	neptsValid	currentCondition
1	1234567890	Male	2001-04-11	1	0194143449	1	1	I am good
2	1234567891	Male	1985-04-10	2	0194143449	7	1	I need nepts
3	1234567892	Male	1985-02-15	1	0194143449	9	1	I need nepts
4	1234567893	Female	1984-10-03	2	0194143449	10	1	I need nepts
5	1234567894	Male	1990-10-18	3	0194143449	11	1	I need nepts
6	1234567895	Male	1990-10-26	1	0194143449	12	1	I need nepts
7	1234567896	Male	1990-06-09	2	0194143449	13	1	I need nepts
8	1234567897	Female	1990-06-09	2	0194143449	14	1	I need nepts
9	1234567898	Female	1990-06-28	1	0194143449	15	1	I need nepts
10	1234567899	Male	1990-09-29	1	0194143449	16	1	I need nepts
11	1234567810	Male	1966-09-09	1	0194143449	17	1	I need nepts
12	1234567811	Male	1966-05-20	1	0194143449	19	1	I need nepts

Figure 36. Database – Patient Table

5.5.1 ER Diagram

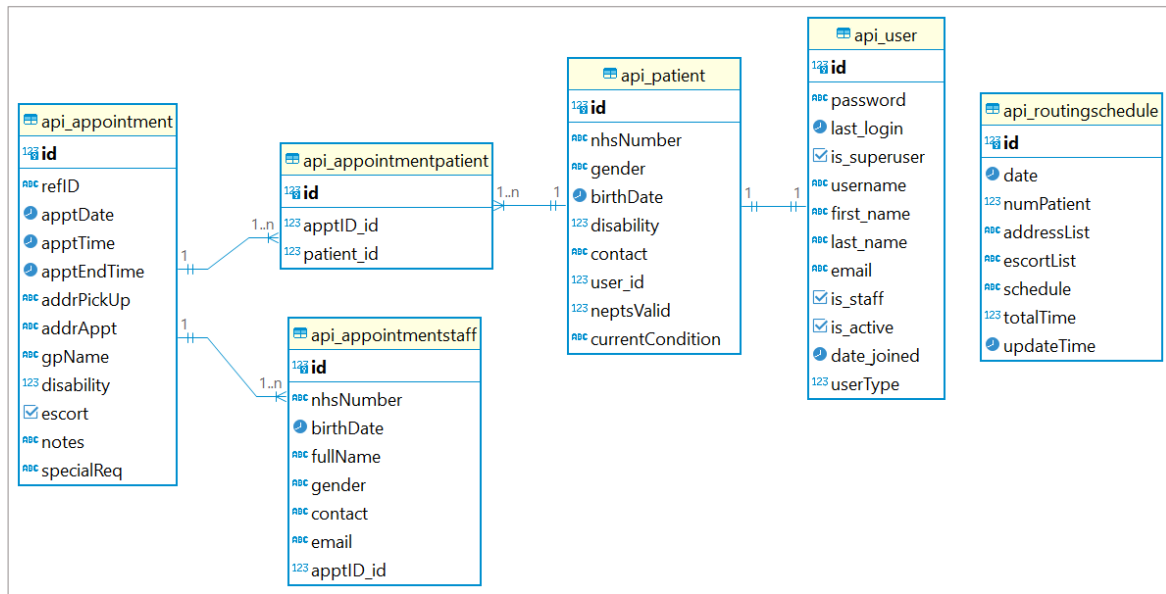


Figure 37. Entity Relationship Diagram

In Figure 37, the entity-relationship diagram illustrates the tables within the database. To enhance simplicity and readability, each table is explained individually in a step-by-step manner, with each one associated with a unique id serving as its primary key.

- (1) **`api_user`**: User table stores all user data for the application, including both staff and patients. While staff members do not currently have a dedicated table for personal details, it is a consideration for future enhancements. The 'userType' attribute distinguishes between patient (userType: 3) and staff (userType: 0). This table maintains a **one-to-one** relationship with the Patient table, ensuring that data deletion in one table triggers deletion in the other, preserving data integrity.
- (2) **`api_patient`**: Patient table that has been mentioned several times in previous sections. It features a foreign key 'user_id' linking to user information in the User table, such as first_name, last_name, and email, which are common attributes for both staff and patients. This design maximizes database capacity utilization. Additionally, it maintains a **one-to-many** relationship with the AppointmentPatient table, meaning each patient can have multiple associated appointments.
- (3) **`api_appointmentpatient`**: AppointmentPatient table stores the appointment information created by patients. It comprises two foreign key fields: 'patientID', linked to the Patient data for patient information, and 'apptID', linked to the Appointment table for appointment details. It maintains a **many-to-one** relationship with the Appointment table, indicating that a patient can create multiple appointments, with each appointment linked to a specific entry in the Appointment table.
- (4) **`api_appointmentstaff`**: AppointmentStaff table stores appointment details manually added by staff, including temporary information for unregistered patients. Similarly to the AppointmentPatient table, it maintains a **many-to-one** relationship with the Appointment table.
- (5) **`api_appointment`**: Appointment table combines entries from both AppointmentPatient and AppointmentStaff. It serves as the repository for all core appointment information in the

application, while the two preceding tables act as ‘bridges’ linking appointment and patient data. This design ensures the efficiency of data retrieval for route scheduling purposes.

- (6) **`api_routingschedule`**: RoutingSchedule operates independently, it does not maintain any relationships with other tables. It serves as the repository for generated routing schedules, eliminating the need for repeated calculation by the server. This approach significantly alleviates the server resource usage and contributes to a smoother user experience.

Chapter 6

6 Implementation of Optimization Routing Algorithm

The optimization routing algorithm is designed to solve the Vehicle Routing Problem (VRP) with multiple constraints, as discussed in Chapter 2. Before delving into its explanation, it is crucial to understand why it is so important. Consider a scenario where there are 5 vehicles tasked with fulfilling 100 patient appointments in a day, some of which have the same appointment times or locations. Without considering other constraints, scheduling just the 5 vehicles could already take hours. Not to mention, if there are more than 5 vehicles and we must ensure patient does not wait excessively for the service. It is almost impossible for a human to solve such complex problems within a limited time.

This is where the algorithm steps in, it gathers all patient appointment information, utilizes genetic algorithm metaheuristics for calculations, and within seconds, an optimized schedule is generated. While it may not always yield the optimal route, it significantly expedites the entire process, saving considerable time.

However, addressing such complex problem could have taken years to develop a working algorithm that fits into to this case. But with the aid of various tools and libraries from Google, I managed to develop it within just three months, as detailed in this chapter.

6.1 Google OR-Tools

Google OR-Tools is a powerful open-source software suite developed by Google, offering a comprehensive set of optimization algorithms and tools. Specifically designed to tackle various types of combinatorial optimization challenges, including the vehicle routing problem. It serves as the backbone of the optimization routing schedule in this application.

6.2 Algorithm Structure

To enhance readability, the algorithm is explained in a clear and structured step-by-step format, it is recommended to look at the code as you read through it (the code is well commented).

(1) Create Data Model

First, the appointment data is cleaned and passed into the function called `‘create_data_model’`. This function organizes the data into a dictionary format, including information such as API key for Google Maps, depot location (starting point for routes), number of vehicles, and their respective capacities, etc.

Then, the time matrix for all addresses is calculated and generated with the help of Google Maps API. This matrix serves as an important variable that defines the travel time between each location nodes.

(2) Create Routing Model

Next, a routing model instance named ‘routing’ is created. It will be used to define constraints, objectives, and solve the routing problem using the optimization algorithms provided by the OR-Tools library.

(3) Add Time Windows Constraint

Time windows defines when each patient is available for pick up. Before setting the rules for this constraint, a time dimension is added to the routing model. Then, the time windows constraint for each vehicle and depot is configured using the time dimension.

(4) Add Pickup Delivery Constraint

Pickup delivery defines the node pairs of patients pick up location and their respective appointment location, where the rules are set to patient must be picked up before being transported to the medical centre.

(5) Add Vehicle Capacity Constraint

Vehicle capacity defines the seats available in each vehicle. The capacity of each vehicle is added to the routing model using a capacity dimension.

(6) Obtain the Schedule

Finally, when all the constraints and rules are added into the routing model, the routing schedule is calculated using the ‘first solution strategy’ with ‘parallel cheapest insertion’ provided by Google OR-Tools. The result is then returned as an HTTP response to the client side.

6.3 Vulnerabilities

While the algorithm performs effectively with the test sets outlined in this report, it is important to acknowledge that it may not function reliably in all scenarios; there is a risk of failure if the problem is overly complex. For instance, the algorithm cannot accommodate calculations involving more than four vehicles with identical capacities. Due limited time for development, the algorithm is implemented without further modifications at present.

Chapter 7

7 Conclusions and Further Work

This report outlined how the proposed web application addresses the problems discussed in Chapter 2. Notably, the application serves as an invaluable educational resource by providing users seamless access to a plenty of comprehensive healthcare information. Through intuitive design and user-friendly interfaces, the patient zone facilitates effortless booking and management of appointments, significantly reducing the administrative burden on both patients and healthcare service providers. Similarly, the staff zone offers a robust solution to scheduling complexities through the implementation of optimized routing schedules, thereby enhancing operational efficiency and resource utilization.

Moreover, the application's robust backend architecture, meticulously designed database structure, and well-crafted API framework ensures a seamless and responsive user experience. By effectively managing data transmission and processing, these foundational elements significantly contribute to the application's reliability and scalability.

In concert with these features, the application embodies a holistic approach to addressing the identified challenges. By seamlessly integrating educational, administrative, and operational functionalities, it not only mitigates existing pain points but also anticipates and accommodates future needs within the non-emergency patient transport services (NEPTS) domain.

However, there are several areas that require further improvement. First, responsiveness needs enhancement as it is not fully configured due to time constraints, potentially impacting user experience on different devices. Second, the booking process could be refined by implementing restrictions that prevent users from booking when no slots are available for the specified day, ensuring efficient appointment management.

Third, to enhance security, implementing an automatic logout feature after 15 minutes of inactivity would safeguard user accounts. Furthermore, in the staff zone, introducing different access levels for staff members based on training and seniority would allow only qualified personnel to validate user eligibility and access to routing schedules.

Finally, optimizing the optimization routing algorithm to accommodate various scenarios would significantly enhance the overall efficiency of the application.

Appendix A

Project Management

Initially, first few weeks was quite stressful, especially during the challenge week. Most students have already chosen their topic and conducted research during the summer vacation, but as a transfer student, I had to finish all these works in a single week. Despite the initial challenges, everything has been progressing smoothly thus far.

An Agile approach was adopted to effectively manage the project, with Kanban used to monitor progress and workflow. It was reassuring to meet with my supervisor during weekly meetings, as it makes sure the project was progressing in the right direction.

A.1 Project Progress

The project's progress can be measured by tracking the completion of issues. Figure A.1 illustrates the cumulative completion of issues throughout the project duration, where purple represents ongoing tasks and green indicates completed ones. There was a notable gap in the initial weeks thanks to stress faced during the challenge week, substantial progress was achieved early on. The overall trend of the chart shows a steady gradient, indicating consistent and gradual progression of the project over the six-month period.

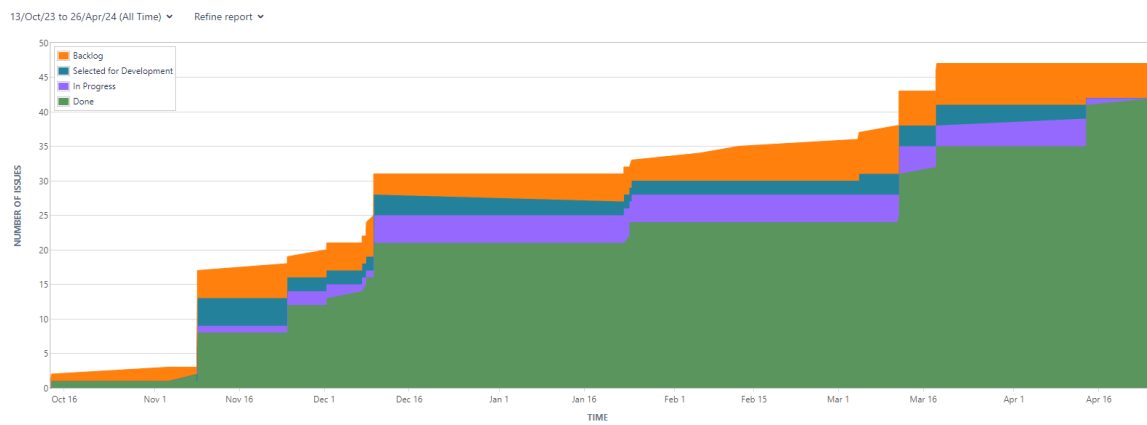


Figure A.1

A.2 What has been learned

In the initial phase of the project, I focused on defining its scope and estimating achievable goals within the six-month timeframe. This involved prioritizing the most critical components for implementation, considering the challenge of completing everything originally planned within the given timeframe. This was particularly challenging since I had not previously developed such a large project independently.

Initially, I had doubts about successfully implementing the routing algorithm into the application, given its complexity and the tight deadline. However, with guidance from my supervisor, I not only overcame these challenges but also gained valuable insights into project management considerations.

Through this experience, I have acquired a deeper understanding of the intricacies involved in managing such projects and have grown more confident in my abilities to navigate complex algorithms and project development.

A.3 Application Deployment

The application is deployed and accessible via this link: <https://tpt-hrap.onrender.com/>. But do note that the trial period for hosting the database and server is limited, so access to the Staff Zone and Patient Zone may be restricted soon. However, if the server is down, you may access to the Demonstration Video available in this [GitHub Repository](#) for an overview of the application's key features.

Bibliography

- [1] Fogue, Manuel, et al. “Non-Emergency Patient Transport Services Planning through Genetic Algorithms.” *Expert Systems with Applications*, vol. 61, Nov. 2016, pp. 262–271.
- [2] Matijević, Luka, et al. “General VNS for Asymmetric Vehicle Routing Problem with Time and Capacity Constraints.” *Computers & Operations Research*, vol. 167, 1 July 2024, p. 106630.
- [3] NHS England.(2021). Improving Non-Emergency Patient Transport Services: Report of the Non-Emergency Patient Transport
- [4] NHS LLR IC. (2022). Report of Findings: Non-Emergency Patient Transport Services (NEPTS) Survey
- [5] Oliveira, João Luiz Alves, et al. “Optimizing Public Transport System Using Biased Random-Key Genetic Algorithm.” *Applied Soft Computing*, vol. 158, 1 June 2024, p. 111578.
- [6] Wassan, Niaz, and Gábor Nagy. “Vehicle Routing Problem with Deliveries and Pickups: Modelling Issues and Meta-Heuristics Solution Approaches.” *International Journal of Transportation*, vol. 2, no. 1, 30 Apr. 2014, pp. 95–110.
- [7] Urban Transport Group, & Community Transport Association. (2017). Total Transport: A Better Approach to Commissioning Non-Emergency Patient Transport?