DRIVECARE CANCER SUPPORT TRANSPORT ORGANISER

Design Document

Student: Adam Lambert
Student Number: C00257510
Supervisor: Dr. Chris Staff
Submission Date: 22/12/2023

Ollscoil Teicneolaíochta an Oirdheiscirt
South East Technological University

Contents

Introduction

The purpose of this design document is to outline the system model of DriveCare. This document will include the technologies used in the creation of this application as well as an overview of the application's structure.

A class diagram will be provided to achieve this overview of the application's structure with sequence diagrams to demonstrate the functionality of the application through the processes it involves.

The database architecture will be discussed with a database diagram to show the chosen structure of the database.

The UI (User Interface) design will also be discussed with prototype screen designs to illustrate the chosen style of the application.

System Design

DriveCare will be created as both a web application and an Android mobile application. The Text Messaging API 46Elks will be used to facilitate the sending and receiving of text messages, which will allow transport coordinators, volunteer drivers and clients to keep in contact about required and organised trips through the application itself. The Routes API will be used to give drivers a visual representation of the most efficient route their assigned trips should follow. Angular will be utilised to create the web applications front end with the back end being created with Python and Django. The mobile application will be created in Android studio using Java. The application will be hosted on Supabase and will use Supabase's Postgres database as information storage.

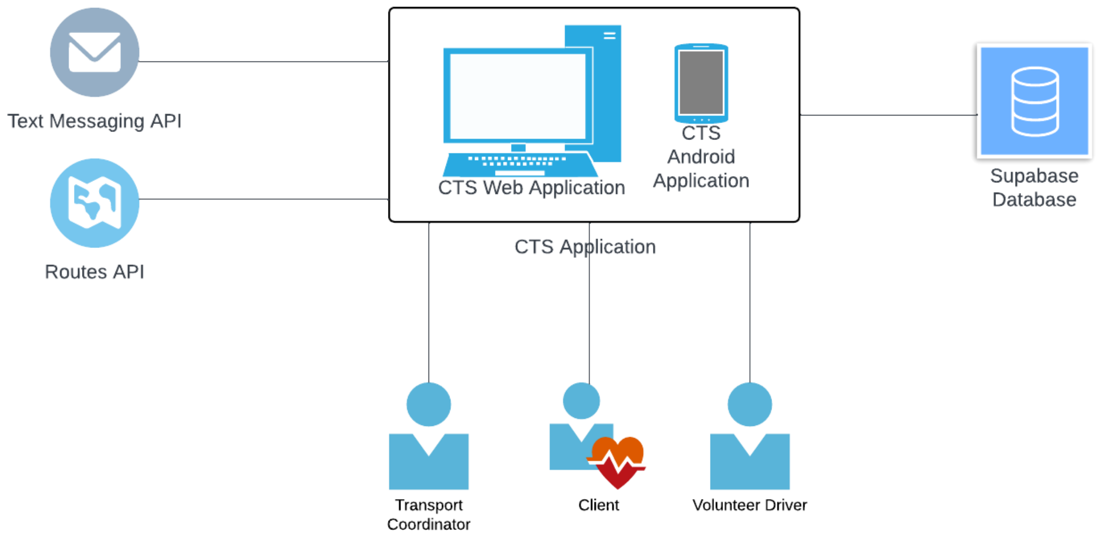


Figure 1. System Diagram

The model system below demonstrates on a high level how the problem of client transportation will be handled within the DriveCare application. This feature is the main value and focus of this system, allowing for clients to have their hospital trips properly organised and scheduled in a way that is convenient for both them and the volunteer drivers.

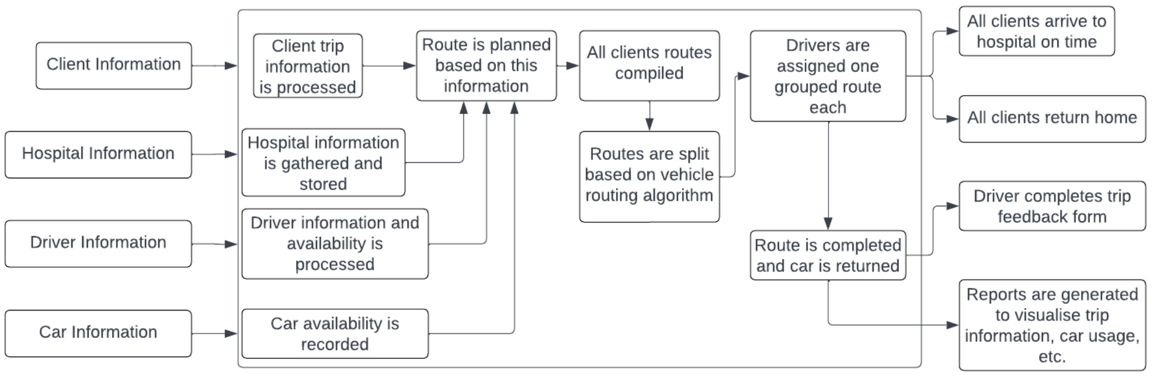

Client Information
Hospital Information
Driver Information
Car Information
Client trip information is processed
Hospital information is gathered and stored
Driver information and availability is processed
Car availability is recorded
Route is planned based on this information
All clients routes compiled
Routes are split based on vehicle routing algorithm
Drivers are assigned one grouped route each
Route is completed and car is returned
All clients arrive to hospital on time
All clients return home
Driver completes trip feedback form
Reports are generated to visualise trip information, car usage, etc.

Figure 2. Model System Architecture Diagram

Critical Algorithm (Vehicle Routing Problem)

One of the most difficult problems to handle in this architecture is the assigning of client routes to drivers. Clients must arrive to their hospital appointments on time, with possibly differing times, pick-up points, and destinations for each client. Drivers must also be considered as, when possible, clients who are picked up from destinations close to each other or share a similar destination should be assigned to the same driver. The number of cars available and space available within these cars also must be considered. This results in a variation of the Vehicle Routing Problem (VRP), with time constraints and a capacity limitation in place. An efficient algorithm in place for this problem would allow for optimal routes being created for each driver that ensure each client's trip requirements are satisfied.

This algorithm would be implemented by storing the locations of all clients, hospitals, and the cancer support office where cars are located. The addresses of these location would need to be geocoded (converted to geographic coordinates) to calculate distances between the locations and map their actual locations for use in the algorithm. The distance each driver would need to travel would then be calculated over a multitude of different possible routes they could take. These possible routes would need a maximum distance to ensure that once a route hits a certain length it is no longer considered viable. One main issue is that this type of algorithm can be incredibly computationally intensive, with a regular VRP algorithm taking only 3 milliseconds with 10 nodes, but 77 years with 20 nodes (Patel, 2023). As this is the case, a meta-heuristic approach will be taken over an exact approach, with a focus placed on generating optimal routes in a reasonable amount of time.

Technologies Used

This section will outline the technologies that this application will be built with. Each section will give a brief overview of the technology chosen, as well as if any alternatives were considered and why it was chosen over these alternatives.

Angular

Angular is a Typescript based, open-source application design framework developed by Google. It will be used to create the front end of the web application. Angular was chosen due to its extensive selection of useful first party libraries that will be beneficial to creating the desired web application.

Django

Django is a python web framework maintained by the Django Software Foundation that will be used to create the web application's back end. Django was selected as it is a robust, versatile, and secure web framework that can be combined with an Angular front end to create a powerful web application, containing built-in guards against many web-based vulnerabilities such as cross-site scripting, adhering to the secure-by-design approach that DriveCare will follow.

Java

Java is an object-oriented, class-based programming language that is primarily used in the creation of Android applications. It will be used to create the Android mobile application of DriveCare.

Supabase

Supabase is a free, open-source Firebase alternative. It will be used to host the DriveCare web application, with its Postgres database being used to store the application's information. Google Firebase was considered in place of Supabase, however Supabase was chosen due to it Postgres database, which allows for the handling of relationships in the database, compared to Firebase's NoSQL database, which cannot. It also contains built-in authentication which is essential with the confidential personal information collected by DriveCare.

46Elks API

46Elks is an SMS, MMS and phone service API that will be used to provide the text messaging service required by DriveCare. Twilio, another communication tool provider was considered for this service, however 46Elks was ultimately chosen as it was more suited to integration with the Python backend of this application with more informative documentation.

Google Routes API

Google Routes is a routing API that provides efficient routing between chosen locations. This service will be used to provide the routing service for drivers using DriveCare. It was chosen as it allows for routes to be calculated through multiple locations, allowing for client's pick-up locations and their desired hospitals to be mapped out in one route.

GitHub

GitHub is a version control software that will be used to store the code of this project. It allows for sufficient version controlling as required by this project.

Visual Studio Code

Visual Studio Code is a code editor created by Microsoft. It will be used as the primary code editor for the web application of this project.

Android Studio

Android Studio is Google's official Android integrated development environment. It will be used as the primary code editor for the Android mobile application of this project.

Secure By Design

As DriveCare is a healthcare related application, confidential personal information must be collected about its users. As a result, this collected data must be handled in a secure and safe manner. This will be assured by a Secure by Design development pattern. Security design will be considered as soon as development begins, with an emphasis on placed on robust security throughout the entire development process. Attacks are assumed to be expected, not just a possibility, resulting in an application that is as secure as possible from the start.

This is achieved in a multitude of ways. The use of Django as the back end of the application allows for immediate built-in protections against some of the most common forms of website vulnerabilities. Supabase provides built-in user authentication, which is imperative to ensure that only valid users of the application have access, and to ensure that each user only has access to what is strictly necessary to their user experience.

The collection of this personal information must also follow the guidelines enforced by GDPR. DriveCare's GDPR compliance is one of the most essential parts of the project because of this. All highly sensitive information in the Supabase database will be encrypted using pgsodium, a plugin which allows for the necessary encryption and decryption features that this project requires. Information will also be stored only as long as is necessary in accordance with GDPR.

Agile Development Methodology

DriveCare's development will follow an Agile Development methodology. Agile software development refers to an iterative process of development, with continuous feedback and reconfiguration of the application in accordance with this feedback. This development process is a much lighter approach than the traditional waterfall method of software development, with focus put on continuous short iterations to refine the product as the process continues.

An agile development method is essential to ensure the proper implementation of the secure-by-design principle discussed above. Constant iterations that produce a testable product will allow for security flaws to be located and removed as the application grows.

A Gantt chart will be created to designate the workflow of the application's development. This will allow for a visualisation of the project's progress and will improve the structure of the development process, with clearly visualised goals to be achieved in iterations.

Detailed Use Cases

This section provides a more detailed description of the Use Cases involved in this project.

Use Case Name: Login
Actors: Transport Coordinator, Volunteer Driver, Client, System Database
Pre-Conditions: The actor involved has a valid account for the application.
Main Success Scenario: 1. The actor involved opens the application. 2. They fill in their login information. 3. They log in to the application.
Post-Conditions: The actor involved is logged in to the application on their account.
Alternative Scenarios: 3(a). The login information is incorrect, and the actor is prompted to enter their details again.

Use Case Name: CRUD Driver
Actors: Transport Coordinator, System Database
Pre-Conditions: The application displays options to Create, Retrieve, Update or Delete a driver account.
Main Success Scenario: 1. The transport coordinator selects the desired option to create, retrieve, update, or delete a driver's account. 2. Depending on the transport coordinator's selection, they enter the driver's details, view the driver's details, edit the driver's details, or delete the driver's account. 3. The transport coordinator confirms their action. 4. The database is updated to reflect the changes made.
Post-Conditions: The driver's account should be modified as per the transport coordinator's modifications.
Alternative Scenarios: 3(a). The transport coordinator cancels the action and is returned to the driver details screen.

Use Case Name: CRUD Client
Actors: Transport Coordinator, System Database
Pre-Conditions: The application displays options to Create, Retrieve, Update or Delete a client account.
Main Success Scenario: 1. The transport coordinator selects the desired option to create, retrieve, update, or delete a client's account. 2. Depending on the transport coordinator's selection, they enter the client's details, view the client's details, edit the client's details, or delete the client's account. 3. The transport coordinator confirms their action. 4. The database is updated to reflect the changes made.
Post-Conditions: The client's account should be modified as per the transport coordinator's modifications.
Alternative Scenarios: 3(a). The transport coordinator cancels the action and is returned to the client details screen.

Use Case Name: Schedule Trip
Actors: Transport Coordinator, System Database
Pre-Conditions: The application displays the option to schedule a trip and a client's transport details have been received.
Main Success Scenario: 1. The transport coordinator selects the option to schedule a trip. 2. The transport coordinator enters the details of the client's transport into the displayed form. 3. The transport coordinator submits the completed form. 4. The database is updated with the new trip information. 5. The calendar is updated with the new trip information.
Post-Conditions: The new trip should be recorded in the database and on the calendar.
Alternative Scenarios: 3(a). The transport coordinator cancels the action and is returned to the schedule trip screen.

Use Case Name: Generate Report
Actors: Transport Coordinator, System Database
Pre-Conditions: The application displays options to generate a report.
Main Success Scenario: 1. The transport coordinator selects the option to generate a report. 2. The transport coordinator selects which report they would like to generate, a report of completed trips, drivers, clients, and cars. 3. The transport coordinator confirms the selection made. 4. The relevant report is generated from the database as per the selection made.
Post-Conditions: The relevant report should be displayed to the transport coordinator.
Alternative Scenarios: 3(a). The transport coordinator cancels the action and is returned to the generate report screen.

Use Case Name: View Calendar
Actors: Transport Coordinator, Volunteer Driver, Client, System Database
Pre-Conditions: The application displays the option to view the calendar.
Main Success Scenario: 1. The actor selects the option to view the calendar. 2. The calendar is displayed with relevant information according to which actor selected it, with drivers and clients only seeing information relevant to them and transport coordinators seeing all trips scheduled.
Post-Conditions: The calendar should be displayed with all relevant information showing.
Alternative Scenarios: None

Use Case Name: Text Message (Send)
Actors: Transport Coordinator, Volunteer Driver, Client, Text Message API, System Database
Pre-Conditions: The application displays the option to send a text message.
Main Success Scenario: 1. The actor selects the option to send a text message. 2. The actor selects who to send the text message to. 3. The actor writes the text message. 4. The actor sends the completed text message.
Post-Conditions: The completed text should be sent to the selected recipient.
Alternative Scenarios: 4(a). The actor cancels the text message and is returned to the text messaging screen.

Use Case Name: Text Message (Receive)
Actors: Transport Coordinator, Volunteer Driver, Client, Text Message API, System Database
Pre-Conditions: The application displays the option to view text messages.
Main Success Scenario: 1. The actor selects the option to view text messages. 2. The actor selects who to view messages from. 3. The messages received from the selected person are displayed.
Post-Conditions: All messages received from the chosen person are displayed.
Alternative Scenarios: None

Use Case Name: Record Log
Actors: Volunteer Driver, System Database
Pre-Conditions: The application displays the option to send a record a trip log.
Main Success Scenario: 1. The driver selects the option to record a trip log. 2. The driver enters the information about the trip. 3. The driver submits the completed trip log form. 4. The database is updated with the completed form.
Post-Conditions: The completed trip log is stored in the database.
Alternative Scenarios: 3(a). The driver cancels the submission and is returned to the record trip log screen.

Use Case Name: Complete Application
Actors: Volunteer Driver, System Database
Pre-Conditions: The person wishing to become a driver is supplied with a link to the application form to become a volunteer driver.
Main Success Scenario: 1. The person opens the supplied link. 2. The person enters the required information to apply to become a volunteer driver. 3. The person submits the completed application form. 4. The completed application form is stored in the database.
Post-Conditions: The completed application form is stored in the database.
Alternative Scenarios: 3(a). The person cancels the submission and is the form is closed.

Use Case Name: View Route
Actors: Volunteer Driver, Route API, System Database
Pre-Conditions: The application displays the option to view a trip’s route.
Main Success Scenario: 1. The driver selects the option to view a trip’s route. 2. The route API calculates the most efficient route for the selected trip. 3. The most efficient route is displayed.
Post-Conditions: The most efficient route for the selected trip is displayed to the driver.
Alternative Scenarios: None

Use Case Name: Send Transport Request
Actors: Client, System Database
Pre-Conditions: The application displays the option to send a transport request.
Main Success Scenario: 1. The client selects the option to request transport. 2. The client enters the details of their required transport. 3. The client submits the completed transport request form. 4. The database is updated with the completed form.
Post-Conditions: The completed transport request is stored in the database.
Alternative Scenarios: 3(a). The client cancels the submission and is returned to the send transport request screen. **3(b).** The transport request is denied.

Use Case Name: Send Feedback
Actors: Client, System Database
Pre-Conditions: The application displays the option to send feedback. The client has finished their time with the cancer support.
Main Success Scenario: 1. The client selects the option to send feedback on the transport service. 2. The client enters their feedback. 3. The client submits the completed feedback form. 4. The database is updated with the completed form.
Post-Conditions: The completed feedback form is stored in the database.
Alternative Scenarios: 3(a). The client cancels the submission and is returned to the send feedback screen.

Sequence Diagrams

Below are sequence diagrams for the most important functionalities in this system.

View Trip Calendar

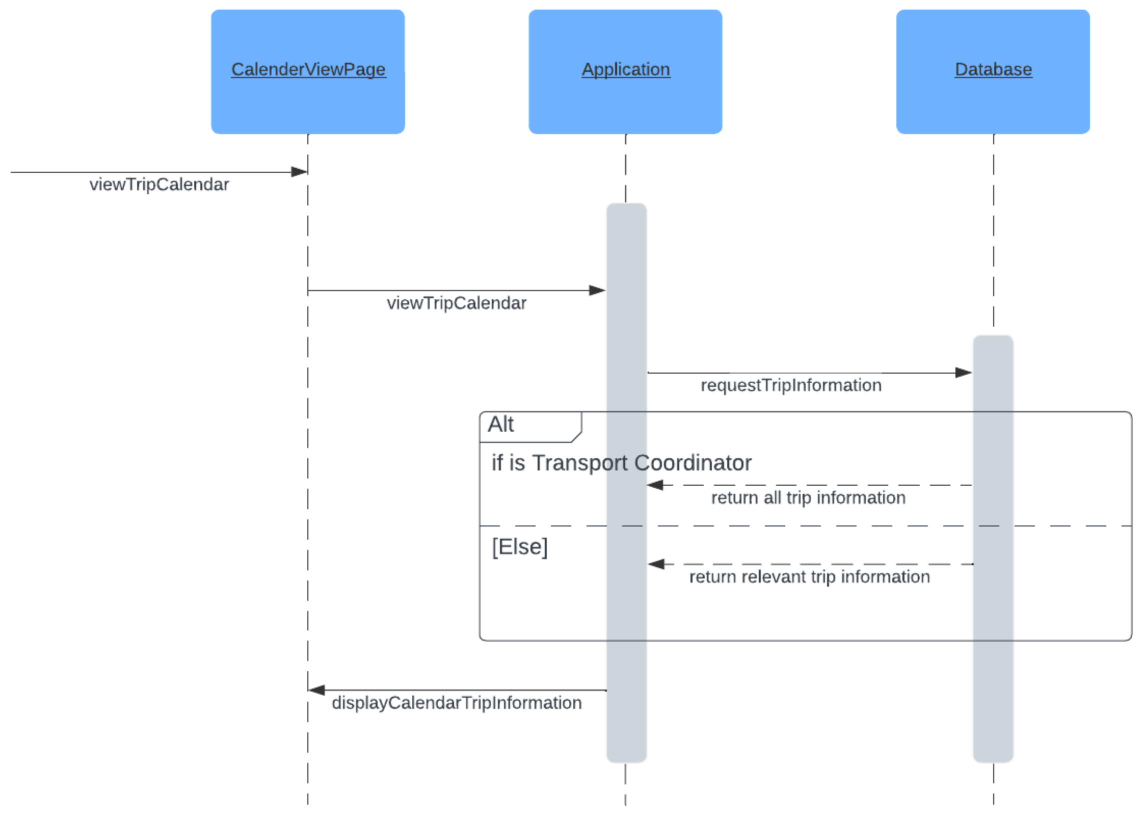

CalenderViewPage
Application
Database
viewTripCalendar
viewTripCalendar
requestTripInformation
Alt
if is Transport Coordinator
return all trip information
[Else]
return relevant trip information
displayCalendarTripInformation

Figure 3. View Trip Calendar sequence diagram

Generate Trip Routes

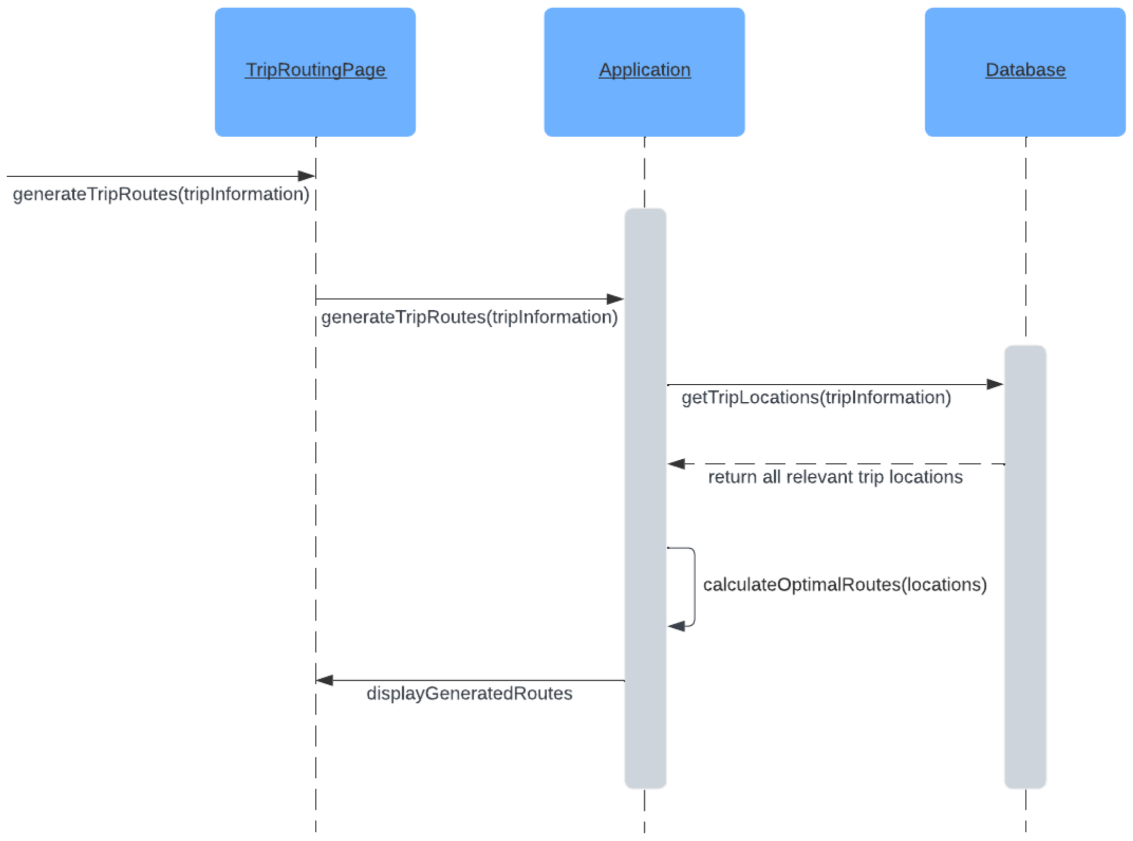

TripRoutingPage
Application
Database
generateTripRoutes(tripInformation)
generateTripRoutes(tripInformation)
getTripLocations(tripInformation)
return all relevant trip locations
calculateOptimalRoutes(locations)
displayGeneratedRoutes

Figure 4. Generate Trip Routes sequence diagram

Class Diagram

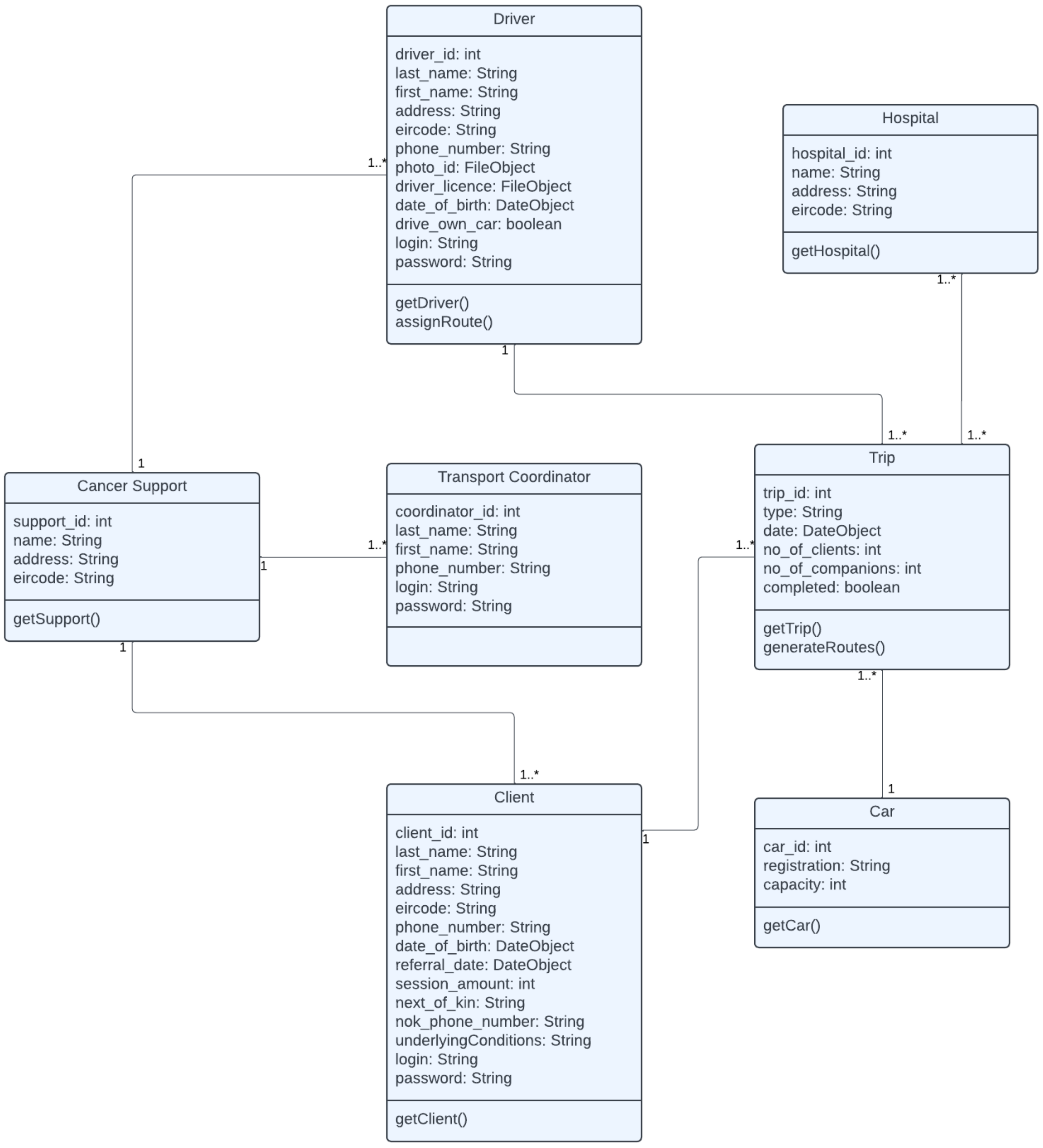


Figure 5. Class diagram

Database Architecture

The database for this project will be built using Supabase following the below Entity-Relationship diagram. The contents of this database will be encrypted to ensure the security of client's confidential information.

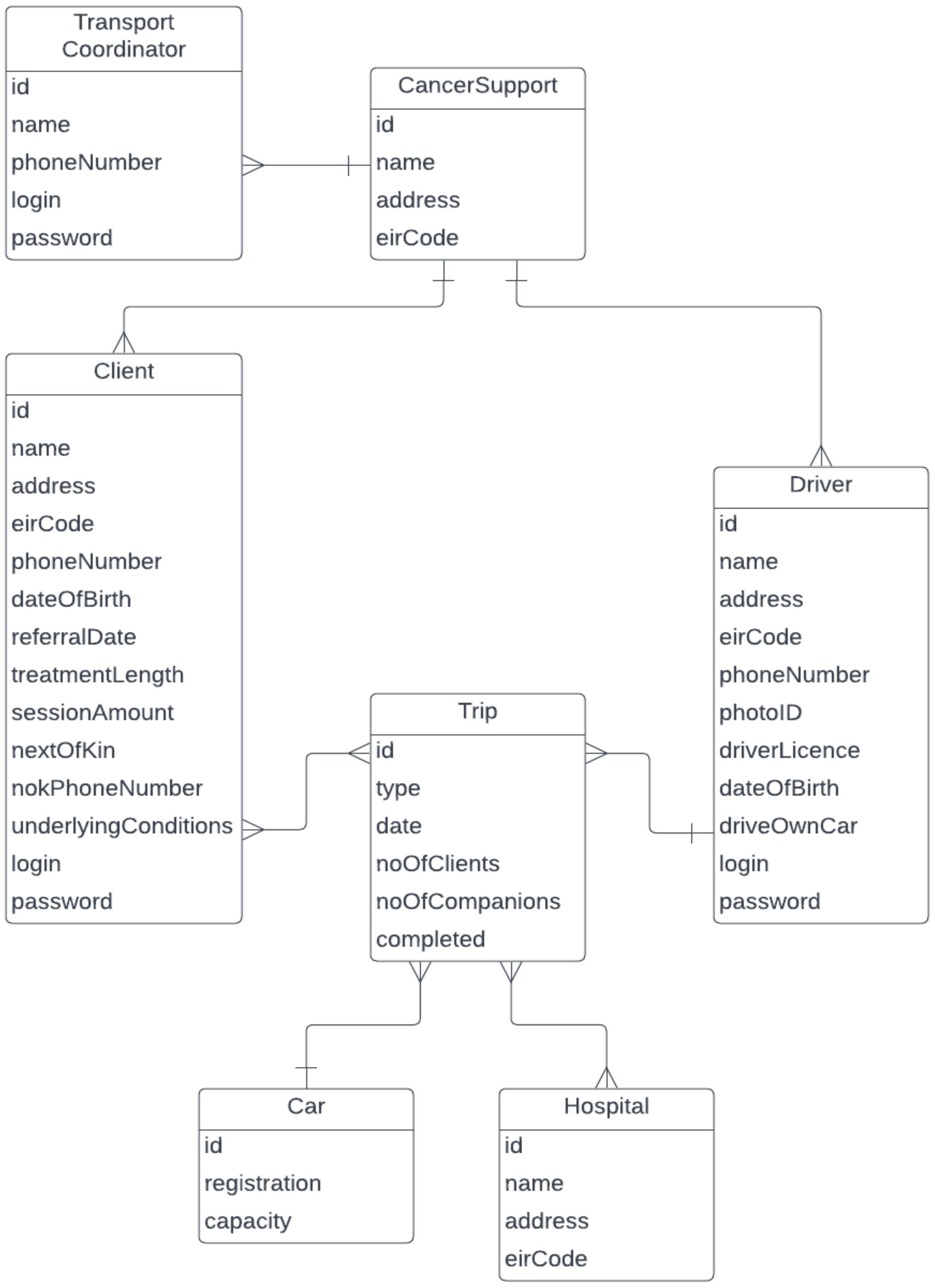
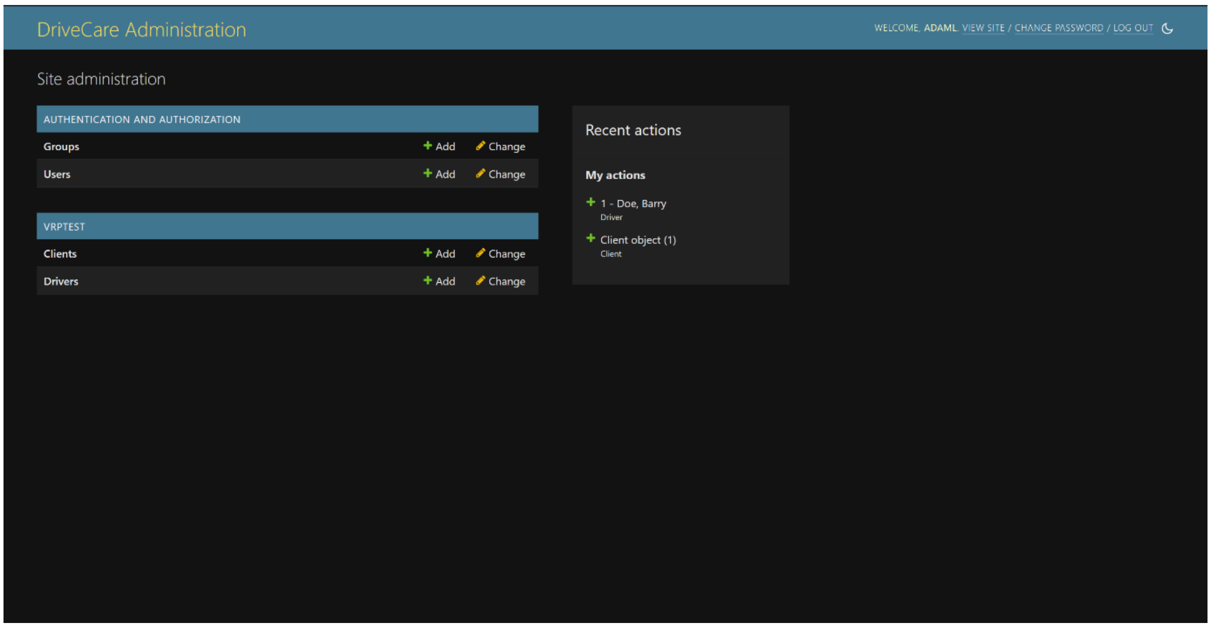


Figure 6. Entity-Relationship diagram

Screen Designs

Admin Home Screen



DriveCare Administration

WELCOME, ADAML. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add	Change
Users	+ Add	Change

VRPTEST

Clients	+ Add	Change
Drivers	+ Add	Change

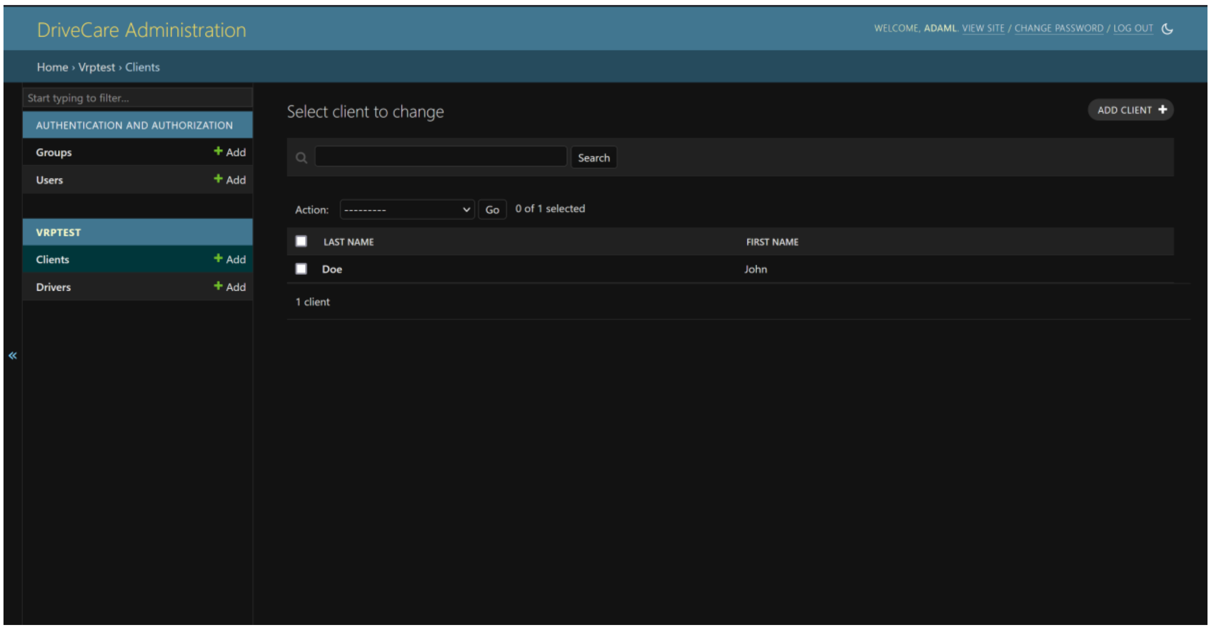
Recent actions

My actions

+ 1 - Doe, Barry
  Driver
+ Client object (1)
  Client

Figure 7. Admin home screen mock-up

View Clients Screen



DriveCare Administration

WELCOME, ADAML. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Vrptest › Clients

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups	+ Add
Users	+ Add

VRPTEST

Clients	+ Add
Drivers	+ Add

Select client to change

ADD CLIENT +

Search

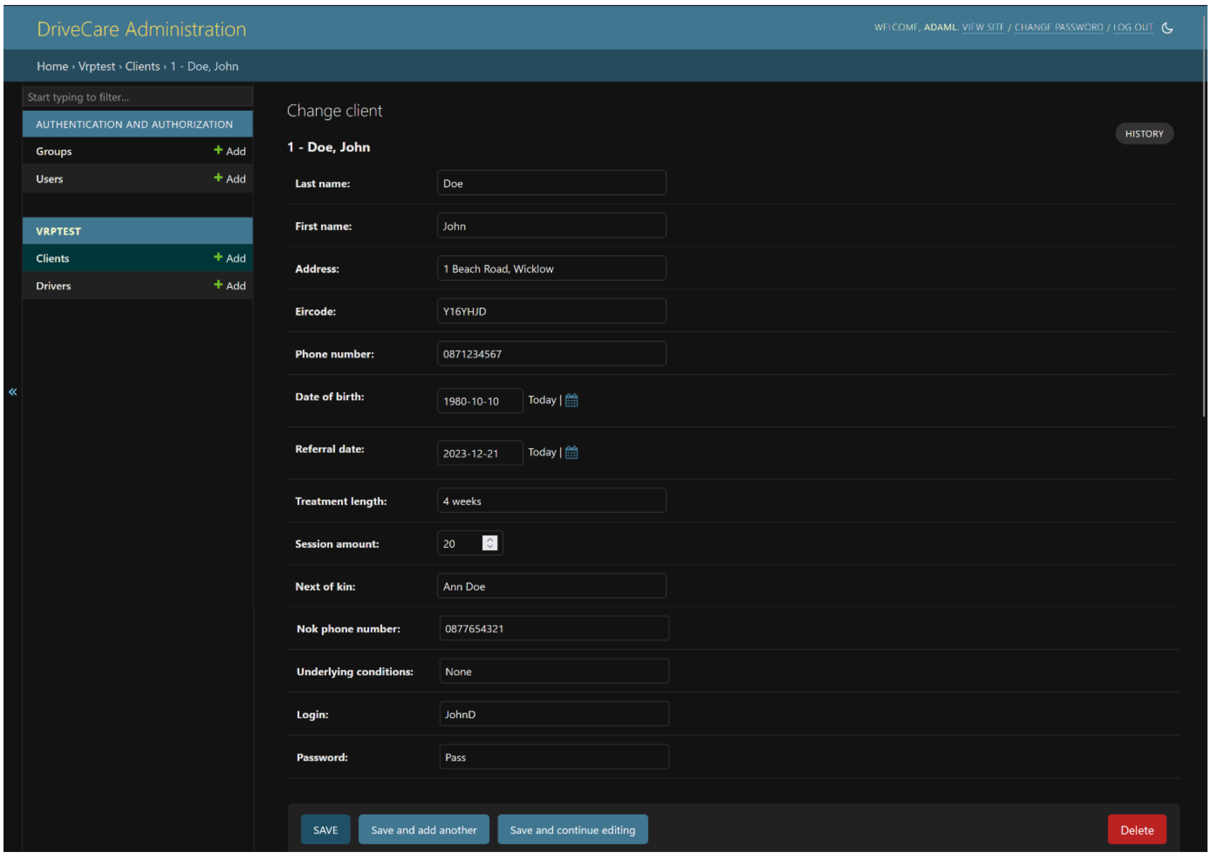
Action: --------- Go 0 of 1 selected

	LAST NAME	FIRST NAME
☐	Doe	John

1 client

Figure 8. View clients screen mock-up

Add Client Screen



DriveCare Administration
WELCOME, ADAML. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Vrptest › Clients › 1 - Doe, John
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
VRPTEST
Clients + Add
Drivers + Add
HISTORY
SAVE
Save and add another
Save and continue editing
Delete

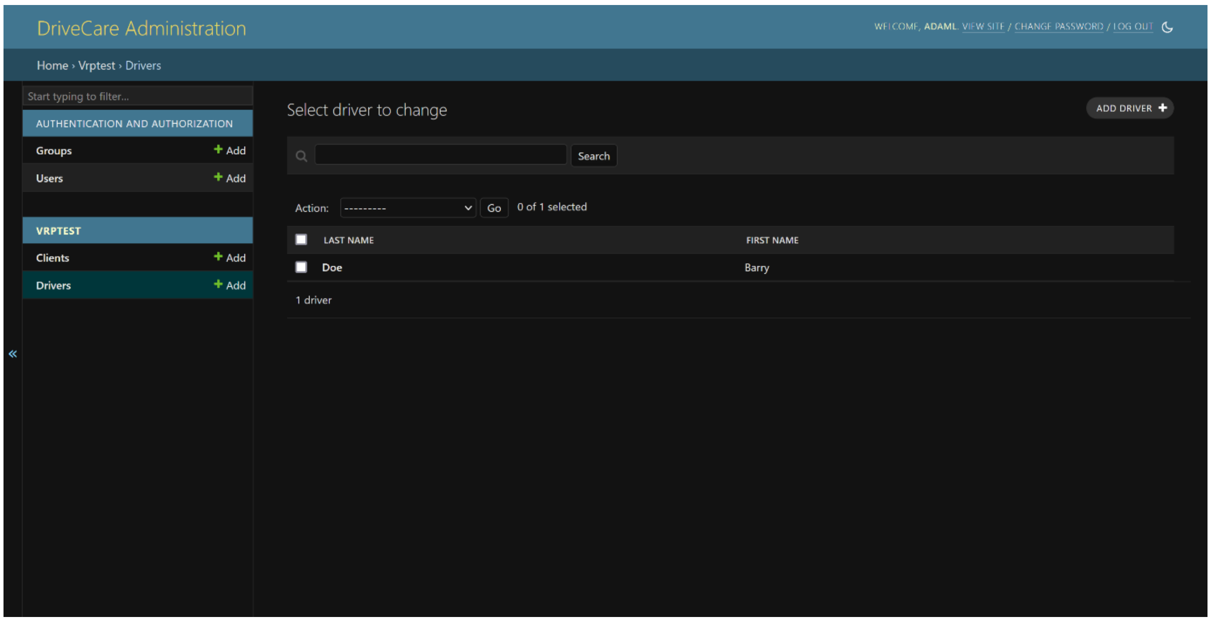
Change client

1 - Doe, John

Last name:	Doe
First name:	John
Address:	1 Beach Road, Wicklow
Eircode:	Y16YHJD
Phone number:	0871234567
Date of birth:	1980-10-10 Today
Referral date:	2023-12-21 Today
Treatment length:	4 weeks
Session amount:	20
Next of kin:	Ann Doe
Nok phone number:	0877654321
Underlying conditions:	None
Login:	JohnD
Password:	Pass

Figure 9. Add client screen mock-up

View Drivers Screen

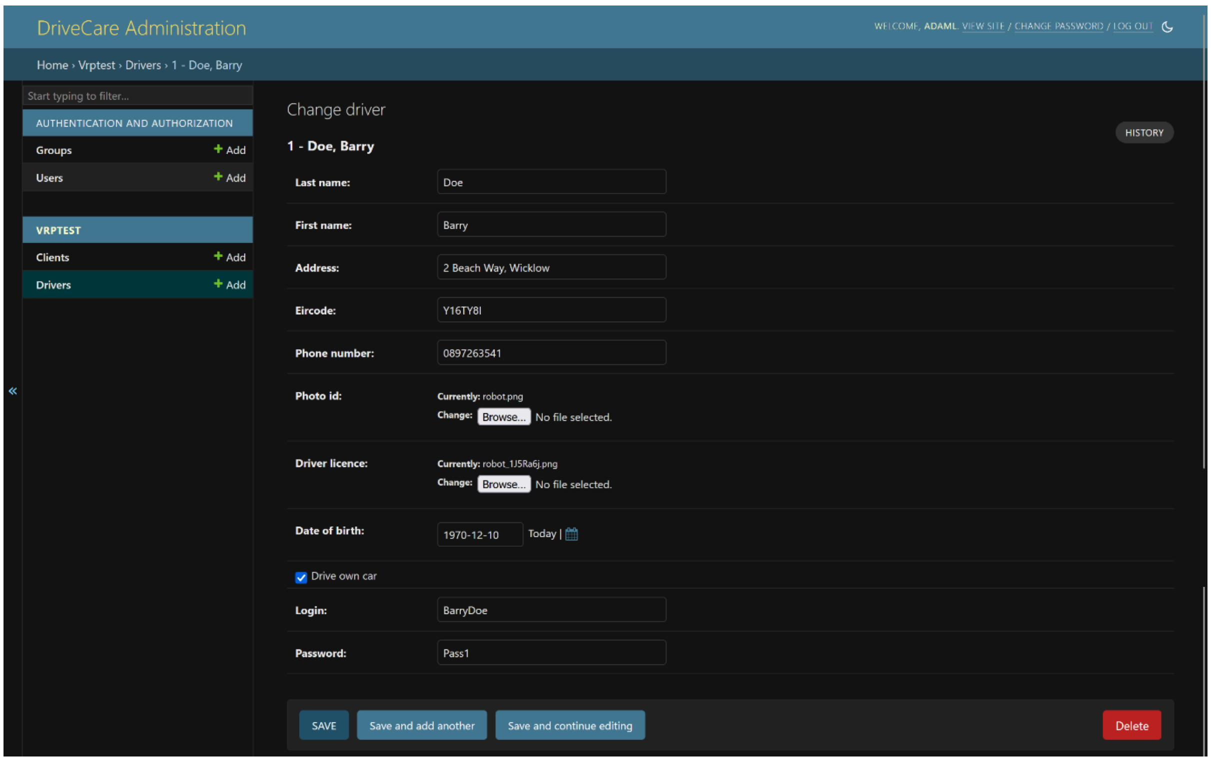

DriveCare Administration
WELCOME, ADAML. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Vrptest › Drivers
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
VRPTEST
Clients + Add
Drivers + Add
Select driver to change
ADD DRIVER +
Search
Action: --------- Go 0 of 1 selected

LAST NAME	FIRST NAME
Doe	Barry

1 driver

Figure 10. View drivers screen mock-up

Add Driver Screen



DriveCare Administration

WELCOME, ADAML. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Vrptest › Drivers › 1 - Doe, Barry

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups + Add

Users + Add

VRPTEST

Clients + Add

Drivers + Add

Change driver

HISTORY

1 - Doe, Barry

Last name:	Doe
First name:	Barry
Address:	2 Beach Way, Wicklow
Eircode:	Y16TY8I
Phone number:	0897263541
Photo id:	**Currently:** robot.png **Change:** Browse... No file selected.
Driver licence:	**Currently:** robot_1J5Ra6j.png **Change:** Browse... No file selected.
Date of birth:	1970-12-10 Today |

[✓] Drive own car

Login:	BarryDoe
Password:	Pass1

SAVE | Save and add another | Save and continue editing | Delete

Figure 11. Add driver screen mock-up

UI/UX Design

The user experience is a pivotal feature of DriveCare. As an application built for use by cancer patients attending hospital appointments, it is essential that the user interface is as intuitive as possible. New clients should be able to locate all the information about their trips without any difficulty.

Drivers and transport coordinators must also find the application intuitive. Drivers will be on tight time frames to transport clients to and from their required destinations, meaning they must be able to clearly find their route details, and be able to differentiate between the various trips they may have to make each day. The complicated VRP algorithm must be effectively hidden from users to not overcomplicate the process of understanding routes and trips; users will simply be able to choose to generate new routes which will be handled using information already available in the database, and not require trips to be manually entered.

The UI design of this application will follow the usual design principles, ensuring simplicity consistency using principles such as Gestalt Psychology like proximity, as shown in the mock up screens, where like elements are grouped together or similarity, where like elements are of the same style.

Conclusion

DriveCare will be created as is detailed throughout this document to create an application that is robust and secure to protect client's privacy, while also maintaining consistent and intuitive functionality across all its features. The chosen technologies will be used to implement an application that will allow clients to easily book hospital trips as they always have with cancer supports, removing hassle from an already difficult time for them while maintaining the previous functionality they are used to, and will allow drivers and transport coordinators to manage these requests and fulfil them in a way that is an improvement from the current procedures in place.

References

Patel, R., 2023. *The Vehicle Routing Problem (VRP) Demystified: Solutions for Your Delivery Operations.* [Online]
Available at: https://www.upperinc.com/blog/vehicle-routing-problem/
[Accessed 10 December 2023].