	Active and Assisted Living Programme **AAL-2016 – Living with Dementia**	
Project Number **AAL-2016-049**		

"CARELINK for Dementia suffers and their community"

Deliverable D2.5

Solution Design

Work package:	WP2 – Requirements and Design
Prepared By/Enquiries To:	Philip O'Brien (pobrien@tssg.org) – Waterford Institute of Technology
Reviewer:	
Status:	Final
Date:	31/10/2020
Version:	1.0.0
Classification:	Public

Authorised by:	**Reviewed by:**
Gary McManus	
Gary McManus WIT	<Name> <Organization>

Authorised date: _31_/_10_/_2020

Project Funding Support

This project would not be possible without the support of the Ambient Assisted Living Joint Programme. The work has been promoted under the project CARELINK, AAL-CALL-2016-049 funded by AAL JP, and co-funded by the European Commission and National Funding Authorities of Ireland, Belgium, Portugal and Switzerland.

Partner	Country	National Funding Body	Logo
WATERFORD INSTITUTE OF TECHNOLOGY (WIT)	IRELAND	ENTERPRISE IRELAND	ENTERPRISE IRELAND
INSTITUTO DE DESENVOLVIMENTO DE NOVAS TECNOLOGIAS (UNINOVA)	PORTUGAL	FUNDAÇÃO PARA A CIÊNCIA E A TECNOLOGIA	FCT Fundação para a Ciência e a Tecnologia MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR
U-SENTRIC	BELGIUM	IWT AGENTSCHAP VOOR INNOVATIE DOOR WETENSCHAP EN TECHNOLOGIE	iwt
OPEN SKY DATA SYSTEMS LTD	IRELAND	ENTERPRISE IRELAND	ENTERPRISE IRELAND
AKADEMIE BERLINGEN	SWITZERLAND	FEDERAL DEPARTMENT OF ECONOMIC AFFAIRS, EDUCATION AND RESEARCH EAER	Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra
CREAGY AG	SWITZERLAND	FEDERAL DEPARTMENT OF ECONOMIC AFFAIRS, EDUCATION AND RESEARCH EAER	Schweizerische Eidgenossenschaft Confédération suisse Confederazione Svizzera Confederaziun svizra

Disclaimer:

CARELINK Project Profile

Contract No.: **AAL-2016-049**

Acronym:	Carelink.
Title:	CARELINK for Dementia suffers and their community.
URL:	www.carelink-aal.org
Twitter	@Carelink_AAL
LinkedIn Group	Carelink
Facebook Page	www.facebook.com/Carelink
Start Date:	01/08/2017
Duration:	30 months

Partners

TSSG	WATERFORD INSTITUTE OF TECHNOLOGY (WIT) - [COORDINATOR]	IRELAND
UNINOVA	UNINOVA - INSTITUTO DE DESENVOLVIMENTO DE NOVAS TECNOLOGIAS (UNI)	PORTUGAL
u-sentric design, for people	U-SENTRIC (USE)	BELGIUM
opensky YOUR VISION.REALISED.	OPEN SKY DATA SYSTEMS LTD (OSD)	IRELAND
AKADEMIE Offen für Neues BERLINGEN	AKADEMIE BERLINGEN (AKA)	SWITZERLAND
CREAGY	CREAGY AG (CRE)	SWITZERLAND

Active and Assisted Living Programme
AAL-2016 – Living with Dementia

Document Control

This deliverable is the responsibility of the Work Package Leader. It is subject to internal review and formal authorisation procedures in line with ISO 9001 international quality standard procedures.

Version	Date	Author(s)	Change Details
0.1	DD/MM/YYYY	Philip O'Brien	Table of Content.
0.2	DD/MM/YYYY	Philip O'Brien	Initial draft for review.
0.3	DD/MM/YYYY	Unonova	Hardware updateds
0.4	27/10/2020	Philip O'Brien	Ready for release
1.0	31/10/2020	Gary McManus	Approved version release.

Executive Summary

Objectives

The Solution Design document builds the basis for all research and development work of the project. It documents the system and software architecture to be used for this solution. Each component will be integrated into the overall system architecture.

Results

Table of Contents

TABLE OF FIGURES

1 INTRODUCTION

This deliverable aims to present a description of the complete Carelink solution, describing the software and hardware components and the communication between them. There is also a detailed explanation of the route prediction research that was carried out.

2 ABBREVIATIONS AND ACRONYMS

Abbreviation	Description
PWD	Person with Dementia
IoT	Internet of Things
LoRa	Long Range Radio
BLE	Bluetooth Low Energy
GNSS	Global Navigation Satellite System
LTE	Long-Term Evolution
GPRS	General Packet Radio Services
MQTT	Message Queuing Telemetry Transport
REST	Representational State Transfer
SMS	Short Message Service
UI	User Interface
JWT	JSON Web Token
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning

3 System Architecture

A key philosophy of this project is that the software, while obviously crucial, is simply the medium with which we realise the Carelink vision. The project is about delivering an innovative solution to the problem of location monitoring for dementia sufferers, not about simply delivering software. To this end we make use of open-source software where possible, saving development effort for the areas where real innovation can be realised.

3.1 Overview

3.2 Hardware

The Carelink localisation solution is constrained by the needs of the targeted user group, which include elder people, in most cases with some degree of dementia.

To fulfil those needs, it is necessary to maximize the following requirements:

- Small Form Factor - to enable the integration and adaptation of the hardware to a desirable piece of clouting or personal wearable;
- Extended Autonomy - to provide commodity and reasonable uptime for an extensive usage (from a week up to a month), using ultra low-power hardware;
- Extended Range - resilient to movement and adaptable to changes in the topology of the surroundings;
- High Availability - to ensure critical communications are always received, regardless of the conditions;

One of the hardest trade-offs of a small form factor device is the limitation of the available energy capacity.

In order to maximize the usage of the battery, a solution of adaptable energy profiles was envisioned.

Each energy profile dynamically changes the actuation of the multiple components of the hardware (sensors, communication radios, localisation modules, etc), depending on the conditions of utilization of the device and the profile of the user (the type and safety of the current location, the time of the day, the remaining battery capacity, the available communication networks, and if it is accompanied by a carer).

By maximizing the correlation of the external conditions to the appropriate usage of the components, it is possible to extend the autonomy of the device and at the same time increase the robustness and resilience of the tracking solution.

To maintain a critical, high availability, localisation solution, it was required to use multiple technologies for both the communications and localisation techniques.

For the communications, the use of emerging, low-power, IoT technologies such as Narrow-Band IoT (also known as LTE-NB), LoRa, Bluetooth Low Energy (BLE) and Wi-Fi, resulted in a distinct range of available networks, capable of coverage in different topology environments: urban, rural, outside, and indoor.

Likewise, for the localisation, besides the Global Navigation Satellite System (GNSS) module (compatible with the GPS, GALILEO, and GLONASS constellations), a multilateration approach was also deployed with the resource of the LoRa network, as well as the use of assisted location services (using available Wi-Fi Access Point's data).

If the device loses coverage from a given technology in use, it is capable of independently falling-back to a better suited one, thus ensuring constant availability.

Multiple external factors can influence the communications and localisation technologies (signals interference’s, noise, reflections, delays; different topology environments: urban, rural, outside, indoor; etc).
Mitigation of (almost) all disruptive influences and behaviours of hardware operations required continuous testing in real conditions, and real-time iteration and maintenance of the hardware operations (such as implementing LoRa as a fall-back method for areas with poor NB-IoT and GPS coverage). Surpassing hardware and firmware limitations, and shortcomings of the devices (such as random failures), was only possible with the implementation of a self-recovery solution. In the end, it served to demonstrate the limitations of the Small Form Factor hardware and the future need for a custom designed solution.

The hardware design went through several iterations and is described in detail in Deliverable D3.3 but a description of the second stage iteration will be provided here. This iteration of the prototype was based on board sizes of approximately 55mm x 22mm e.g. Pycom fipy and Sodaq Sara boards

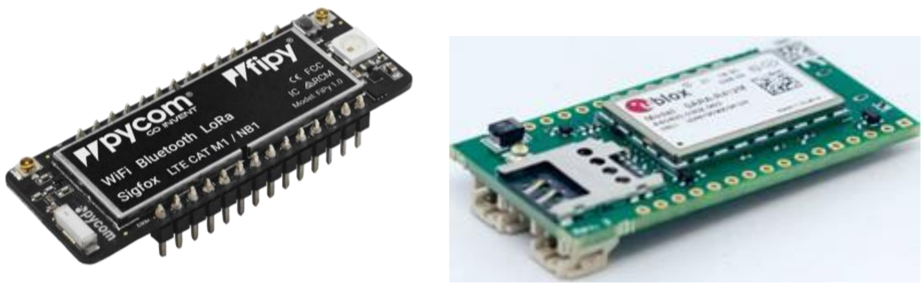


Figure 1 Prototype Boards

The availability of the boards and respective supporting documentation, their size and footprint, technological capabilities and usability in user trials, were the determining factors for choosing the boards to be used in the field trials. Thus, the Pycom Fipy + Pytrack combo and the Sodaq Sara SFF R412M were the chosen boards.

The Pycom combo was chosen due to the greater number of radio technologies, 6, that provide multiple fallback capabilities in case one technology fails. The Sodaq R412M was chosen for having the smallest size footprint, while also having one fallback communication option, from LTE to GPRS.

3.2.1 Energy Management Profiles

To optimize the Carelink device's power autonomy, the configuration of each device's components (communication, tracking and sensors) is modified by the platform, using a set of Energy Profiles that adjust the modes of operation to the conditions of the PwD.

Using the platform as a manager of the energy settings has three major benefits. First, it frees the devices from the processing required to determine the conditions to apply each profile, which by itself already saves a considerable amount of power. Secondly, it enables tweaking the components configurations remotely, in real-time, without the need to recall the devices to update the settings physically. Thirdly, it empowers the Carelink system with the possibility to adjust the Energy Profiles of a device to the respective PwD's usage habits, by pre-emptively adapting the configurations to regular actions, creating a personalization of the system around the PwD needs.

Depending on the conditions the PwD is in, there is a need for a higher or lower interval of time, of the status messages sent between the device and the platform, or from the data polled by the device's sensors. These conditions can be the "Location" of the PwD, the "Time" of the day, if the PwD is "Accompanied" by a carer, and the overall "Wellbeing" of the PwD.

Again, this implementation is described in detail in D3.3.

3.3 Software

Early on we decided we would follow a microservices approach. Microservices are small, autonomous services that work together. Many organisations have found that by embracing fine-grained, microservice architectures, they can deliver software faster and embrace newer technologies. More importantly, however, microservices give us significantly more freedom to react and make different decisions, allowing us to respond faster to the inevitable change that impacts all of us. (Newman, 2015)

Common characteristics of a microservice architecture are

- **Componentisation via services** – treat each service as a component that can be independently deployed
- **Organised around business capabilities** – take a broad stack approach to the implementation of software, including UI, persistent storage etc.
- **Products not projects** – rather than handing over a component for maintenance when complete, the team instead takes ownership of that component for its lifetime
- **Smart endpoints, dumb pipes** – services own their own domain logic and should be as decoupled and cohesive as possible. Inter-service communication should be as simple as possible using simple REST services, or lightweight messaging
- **Decentralised governance** – or use the right tool for the job e.g. .NET/react for the frontend, Python for machine learning etc.
- **Decentralised data management** – services own their own data

- **Infrastructure automation** - closely tied to continuous delivery, microservices are simpler to rest and thus to deploy, since they are highly decoupled
- **Evolutionary design** – facilitates rapid evolution of a system through the replaceable, or even transitory, nature of components. If the services are sufficiently granular and properly decoupled, only select services need to be upgraded/replaced. This is particularly appealing in the context of Carelink as we develop more sophisticated location tracking algorithms.

In the context of a project like Carelink, the benefits of such an approach are numerous, the most notable being that

1. Distributed teams can collaborate more effectively, without being bound by adherence to the same tools/technologies
2. The project can adjust its trajectory more easily, as new data emerges from user trials, and market research

Figure 2 shows the Carelink implementation of a microservices architecture with thirteen distinct services. Integration is obviously paramount when dealing with multiple services. Since communication between services can be both synchronous and asynchronous, we utilise two approaches

- Messaging
  - Message Queuing Telemetry Transport (MQTT) as a messaging protocol
- REST
  - Some services will need to communicate synchronously so transporting data as JSON via REST is the most sensible way to go

When it comes to storing data, two approaches are used

- An immutable data store for end-user data built on Postgres
- Each microservice has the option of also using their own datastore where appropriate

Each microservice is containerised using Docker, and these containers are orchestrated using Docker Compose.

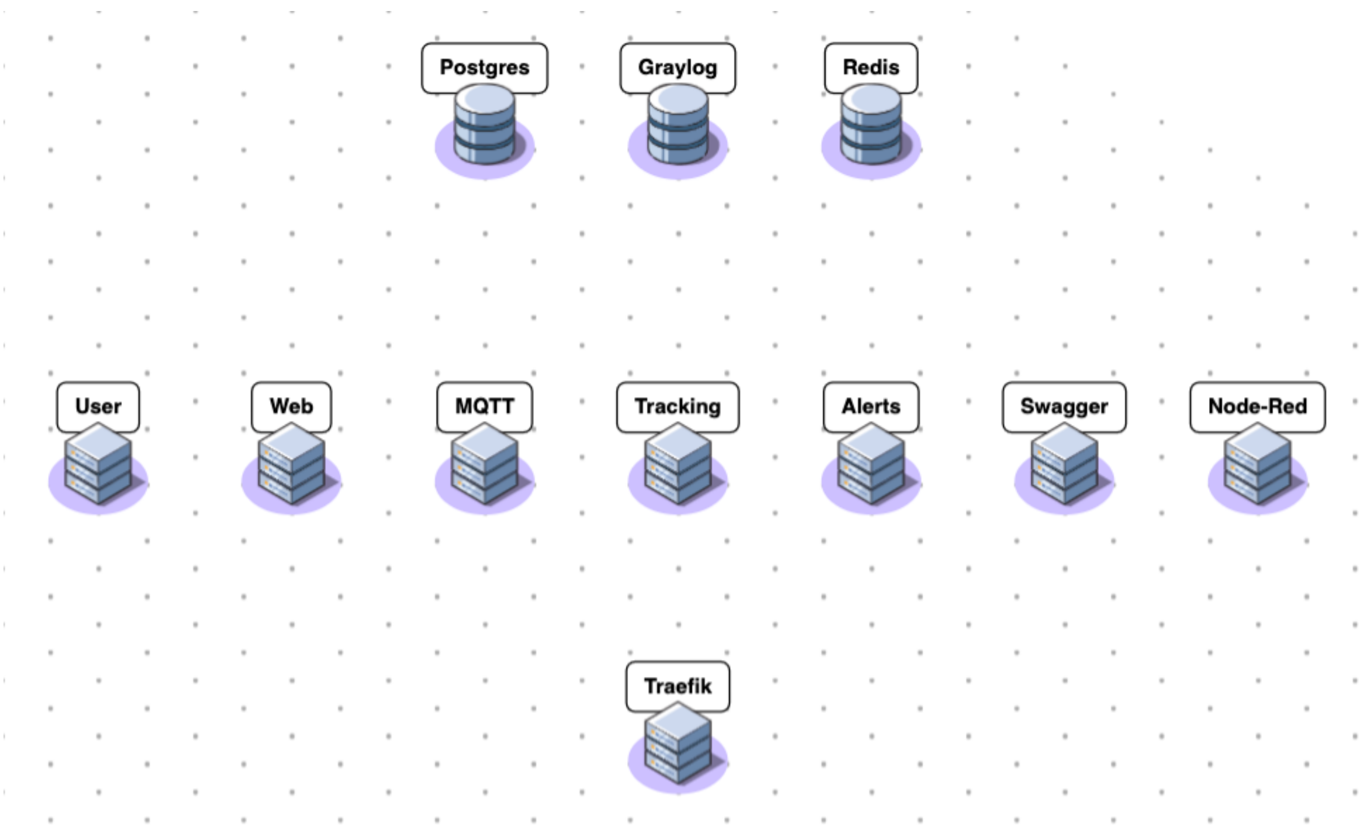


Figure 2 High-Level Platform Architecture

3.3.1 Messaging

A requirement of the Carelink solution was a lightweight messaging protocol due to the fact that maximization of energy consumption can require reducing the load and processing power on the device, using the platform whenever possible. The proposed protocol, MQTT, satisfies these conditions and integrates easily with the platform.

The MQTT specification is detailed in Appendix 2 in D3.3

3.3.2 Data Flow

The typical flow of data through the platform is straightforward

- Location data from a Carelink device is transmitted from the device to the backend platform
- This data is transmitted using the MQTT protocol
- The Tracking service listens for data arriving via MQTT and when this occurs it analyses the location data to see if an alert is necessary
- If an alert is warranted the Tracking Service publishes a message to the Alerts topic in MQTT, which is picked up by the Notification service
- The Alerting service then triggers an SMS notification to the carer of the PwD

3.3.3 The Core Services

3.3.3.1 Web Service

The user interface will be provided by the web service. The primary UI will be a responsive web application, permitting use from a browser, but also useable from devices.

The web service, i.e. the front-end, is described in detail in D3.2 so there is no need to duplicate the description here, although we will include the technology listing so all technologies for the platform can be found in this one document. The important point in the context of the broader platform is the frontend is treated like all the other Carelink microservices, i.e. a self-contained software component that can be developed and deployed independently.

The GUI Interface is built with the following modern technologies:

- HTML5 and CSS3
- Twitter Bootstrap 4 as CSS Framework
- Javscript ES6, ReactJS with Redux
- .NET Core for routing and hosting purposes
- SASS
- Webpack for JavaScript code compile automation

Every time the GUI retrieves/sends information it needs to connect to the API by sending JWT token stored in browser's location storage after successful login process. The UI app is built with HTML, CSS and JavaScript. It uses responsive patterns so that it is a mobile friendly application.

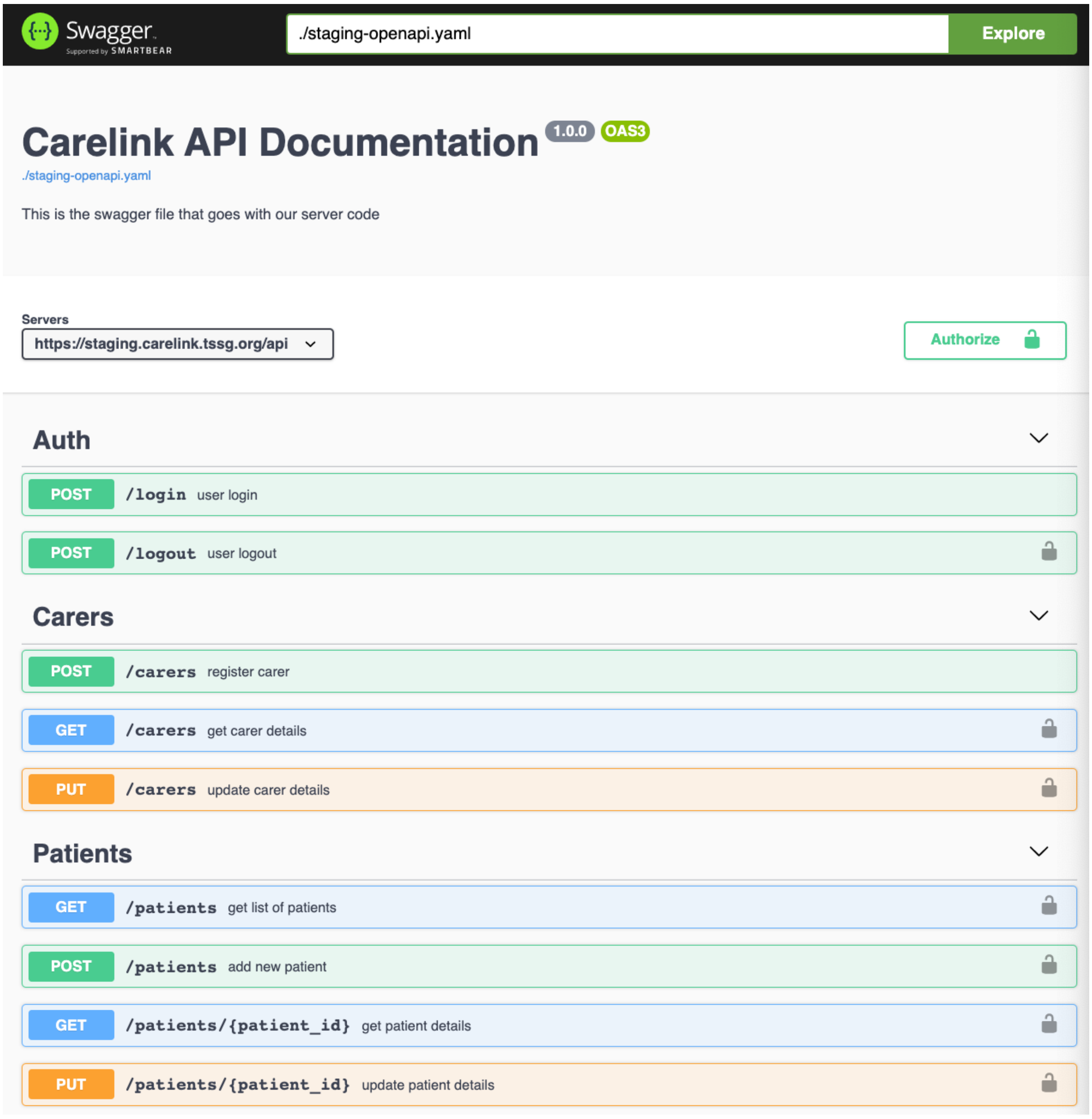
3.3.3.2 User service

The User service exposes an API to facilitate the following

- User registration
- User authentication/authorisation (via JSON Web Tokens as described in section **Error! Reference source not found.**)
- Retrieve/update carer details
- Retrieve/update PwD details
- Retrieve/update Location details
- Retrieve/update Device details
- Retrieve/update Alert details
- Retrieve/update Zone details
- Retrieve/update Route details
- Retrieve/update Energy Profiles

The API is documented using Swagger following the OpenAPI Specification (OAS) v3. The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to RESTful APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection.[1]

Figure 3 shows a screenshot of the online swagger documentation (only some of the API endpoints are visible).



Swagger Supported by SMARTBEAR
./staging-openapi.yaml
Explore

Carelink API Documentation 1.0.0 OAS3
./staging-openapi.yaml

This is the swagger file that goes with our server code

Servers
https://staging.carelink.tssg.org/api
Authorize

Auth
POST /login user login
POST /logout user logout

Carers
POST /carers register carer
GET /carers get carer details
PUT /carers update carer details

Patients
GET /patients get list of patients
POST /patients add new patient
GET /patients/{patient_id} get patient details
PUT /patients/{patient_id} update patient details

Figure 3 User Service Swagger Documentation

[1] https://swagger.io/specification/

3.3.3.3 Tracking Service

As described in Section **Error! Reference source not found.** the Tracking Service listens for messages on particular MQTT topics, and when location data is being transmitted to the platform, it is analysed to detect if that PwD is wandering.

The most fundamental level of analysis focuses solely on whether or not a PwD that is mobile is within a boundary that has been defined to represent a safe zone. This zone(s) is created by the carer and/or the PwD using the web frontend and essentially says "once the PwD is moving in this zone everything can be considered fine, but if they leave this zone an alert must be sent".

Take the example shown in **Figure 2** where the four blue markers represent the boundary of a safe zone. The red marker, representing the location of the PwD, is outside of the zone and so in this instance an alert would have been triggered. This example is simplified just to visualise the concept, such a small safe zone would clearly be draconian in terms of how restrictive it is.

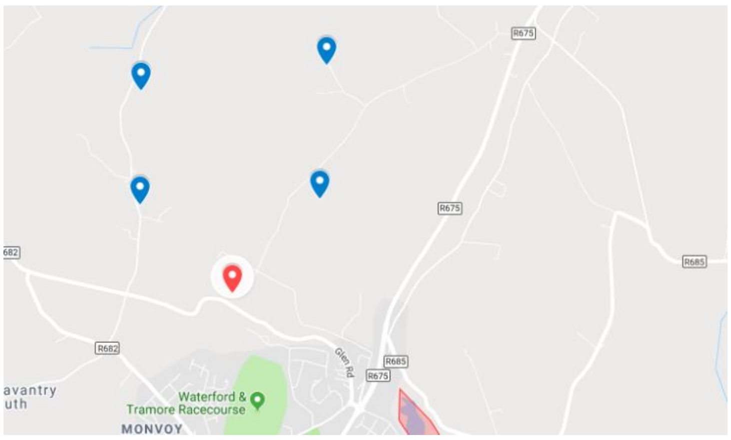
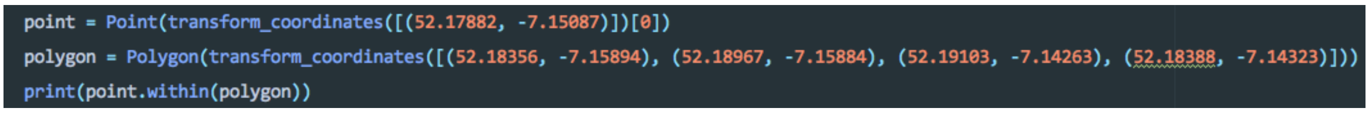


Figure 4 PwD Outside Safe Zone

What at first seems a very easy problem to address, actually has some technical nuances. It is a problem known as the Point-in-Polygon problem, and before the location points can be used, they must first be transformed from the GPS coordinate system (3D) to a cartesian coordinate system (2D). This use of specific geographical projections is vital to ensure accurate analysis. To achieve this we make use of the Python package, Shapely.[2] A snippet of this code is shown in **Figure 5** showing how Shapely allows us to perform both the coordinate transformation, as well as the point-in-polygon detection.

[2] https://pypi.org/project/Shapely/



```
point = Point(transform_coordinates([(52.17882, -7.15087)])[0])
polygon = Polygon(transform_coordinates([(52.18356, -7.15894), (52.18967, -7.15884), (52.19103, -7.14263), (52.18388, -7.14323)]))
print(point.within(polygon))
```

Figure 5 Python Point-in-Polygon Code

3.3.3.4 Analysis Service

There are several different types of analysis that we planned to use to best manage wandering behaviour. The failsafe level i.e. safe-zone detection is implemented by the Tracking Service as described in Section 3.3.3.3.

3.3.3.4.1 Anomaly detection in GPS traces

Figure 6 shows a representation of some of the navigation patterns that have been observed during a wandering event (Martino-Saltzman, Blasch, Morris, & McNeal, 1991).

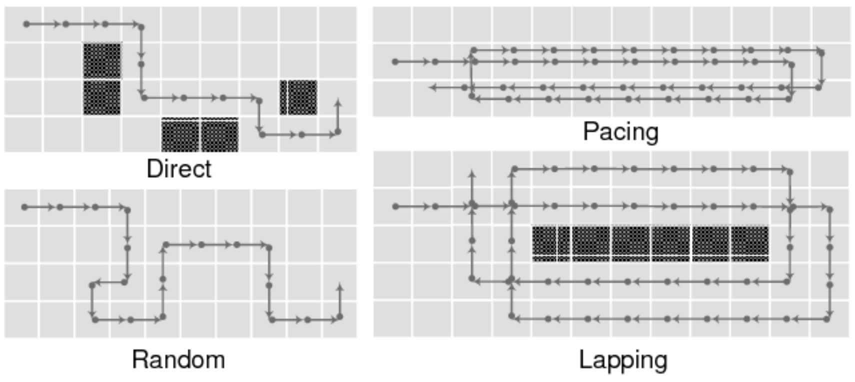
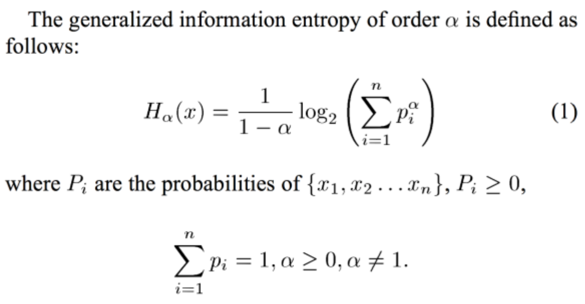


Figure 6 Hallmark Patterns of Wandering

We began developing algorithms to classify the three main wandering patterns i.e. Pacing, Random, and Lapping. The first pattern we decided to address was Pacing, i.e. repeating back and forth along a route. We took inspiration from (Xiang, Li, & Zhou, 2011) in which the authors used the concept of Entropy to detect distributed denial of service (DDoS) attacks. They calculate entropy using **Equation 1**.



The generalized information entropy of order α is defined as follows:

$$H_\alpha(x) = \frac{1}{1-\alpha}\log_2\left(\sum_{i=1}^{n} p_i^\alpha\right) \qquad (1)$$

where P_i are the probabilities of $\{x_1, x_2 \ldots x_n\}$, $P_i \geq 0$,

$$\sum_{i=1}^{n} p_i = 1, \alpha \geq 0, \alpha \neq 1.$$

Equation 1 Generalised Information Entropy

Essentially what this gives is a figure representing the level of unique information in a dataset. In an extremely simplified example comparing two datasets D1, and D2, where D1 = [1, 2, 3, 4, 5] and D2 = [1, 1, 1, 1, 1] the entropy score for D1 would be higher because there is more information represented in the set, rather than repeating values in D2. It calculates the probability of points occurring in a dataset. Those occurring more frequently have less salience.

We realised that by applying this concept to the analysis of GPS data, if the PwD began pacing, then as the GPS points repeat the entropy score should drop.

We tested this against the Microsoft Geolife GPS Trajectory Dataset[3]. **Figure 7** shows a sample trace from this dataset. A heatmap is used to highlight repeating points i.e. as the colour of the trace goes from green to red this indicates that the route is being repeated. So, the pacing section of the route is the red part.

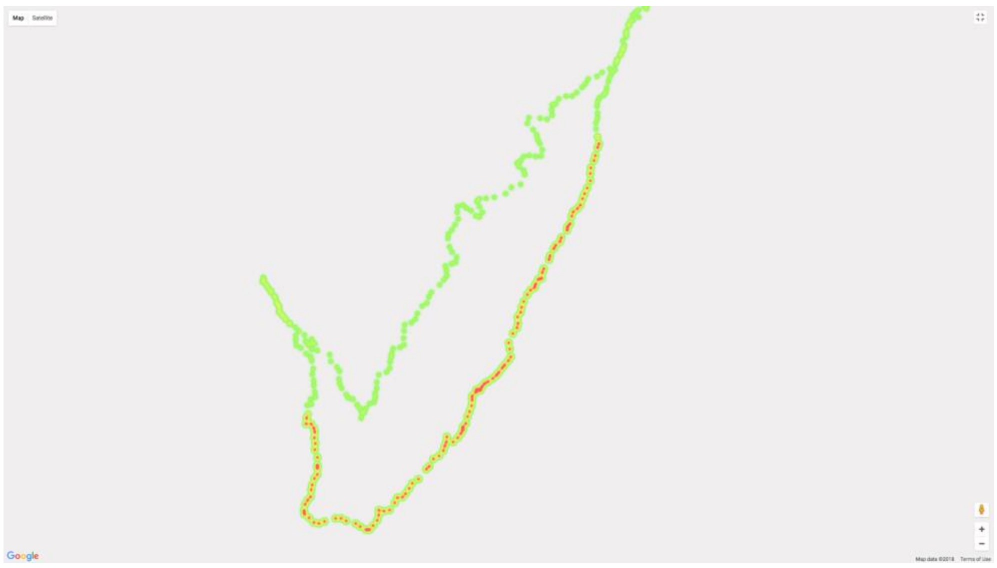


Figure 7 Heatmap showing repeating points as increase in colour intensity

Figure 8 shows the result of our pacing detection algorithm. For each new GPS point reported, the algorithm calculates the entropy score (represented by the blue line in the figure). As soon as the pacing begins the entropy drops and continues to drop until the pacing stops. This pacing correlates to the highlighted region in the figure.

[3] https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/

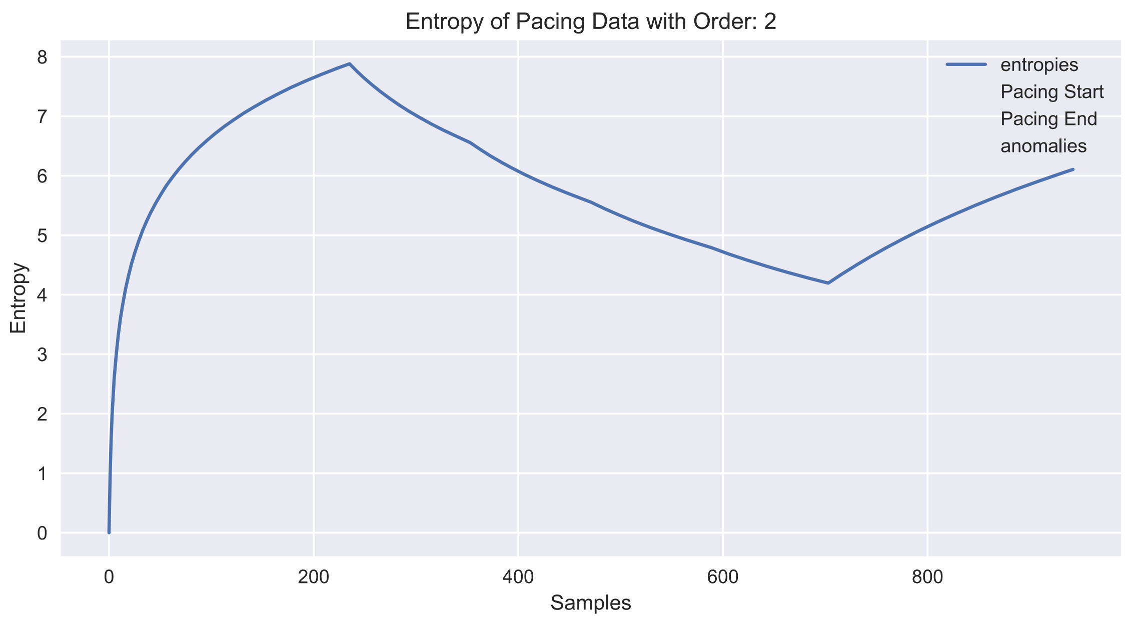


Figure 8 Entropy Calculation for GPS Trace

What this means is that if a PwD begins pacing while walking the algorithm can detect this in real-time and alert if appropriate. Having brought the pacing detection to this stage our plan was to test it further and then develop complementary algorithms to detect both Random and Lapping patterns. At this stage, however, we were also considering the problem of route prediction i.e. rather than just classifying the behaviour of the PwD while they are walking, can we predict their likely destination. Discussing this with the Carelink reviewers there was broad agreement that although this would be a very challenging problem, it had great potential. We then made the decision to direct our focus here rather than continuing the behaviour classification.

3.3.3.4.2 Route Prediction

In the TSSG we had previously applied Deep Learning to sequence prediction for forecasting problems.

It was our belief that a similar DL approach could be very effective in this regard. The thinking was that if the system could learn the common routes they take, it should quickly be able to classify if they are on a route leading to a common destination. If this is not the case, then it is implied that they are not going towards a known destination and so may be wandering. There are obviously many questions to try and address with such a problem but having discussed with the reviewers the potential merit in taking on this challenge we decided to factor it in to our work.

This ML-based sub-system aims to classify the normal routes a PwD usually takes from the routes that they might take when they get lost during a wandering episode. Eventually, predicting the destination a patient is heading to, as early as possible, based on the real-time changes of their location.

The Route Prediction System architecture consists of 5 modules, as shown in Figure 9

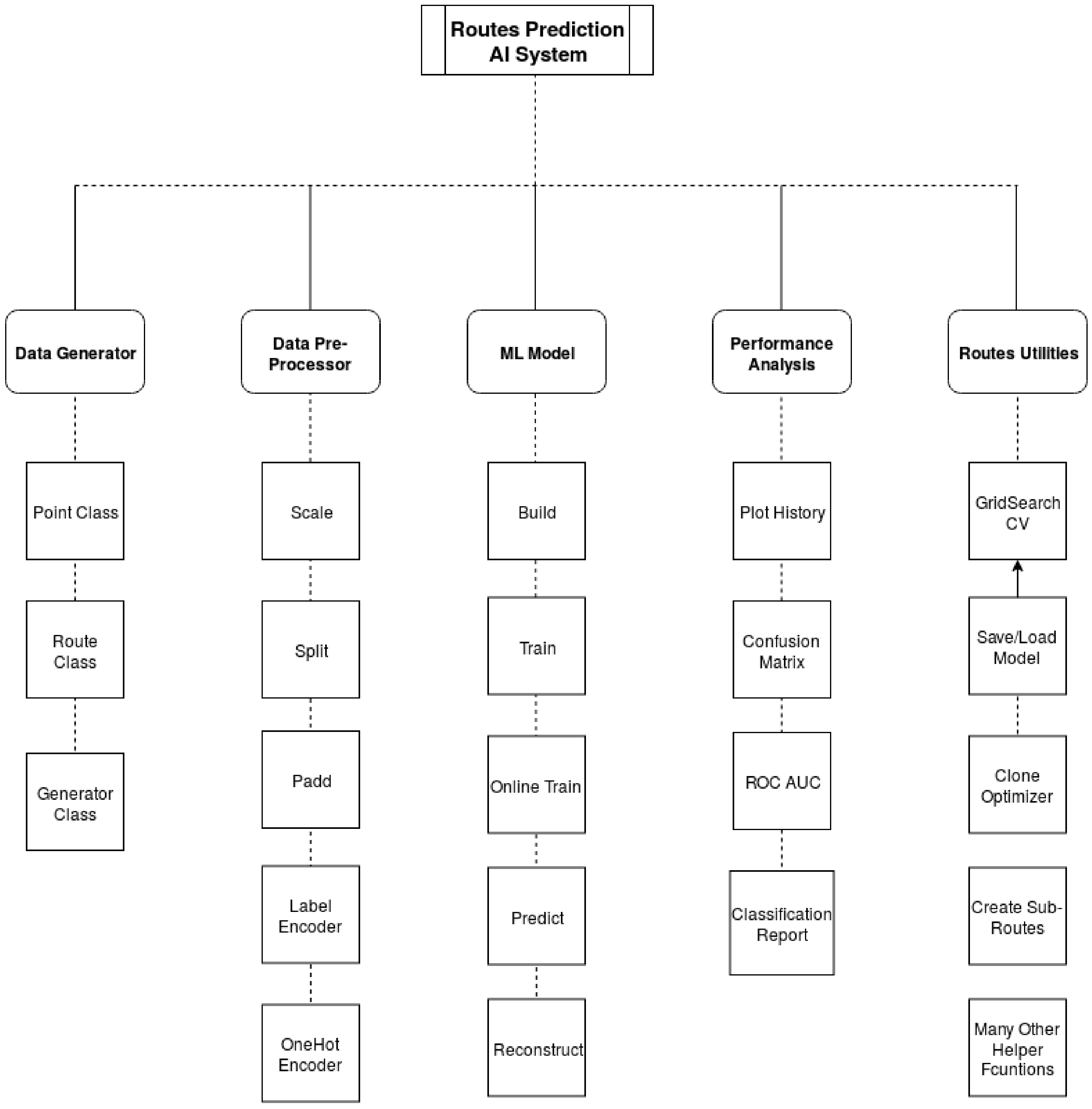


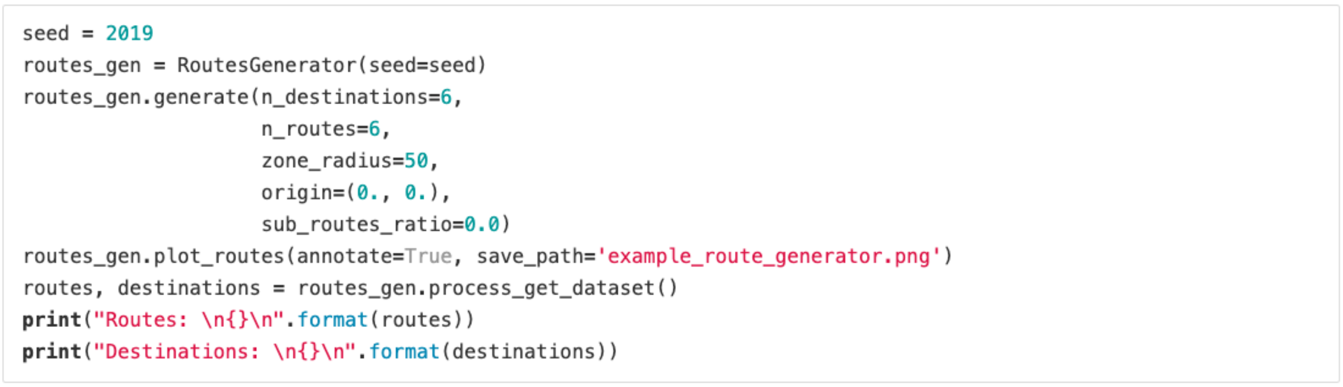
Figure 9 Route Prediction Architecture

Data Generator

Even with the trials that were planned pre-covid we believed we could not record enough real data to experiment with an AI algorithm such as this. So, we decided to generate realistic, synthetic data to test our hypothesis, and the hope was then to capture as much real data as possible during the trials and asses how the algorithm would perform.

This module provides the functionality to generate the highly customisable dataset of simulated routes and destinations taken by a person. A user can specify the number of main routes, the number of sub-routes and the number of destinations, in addition to the radius of the zone those routes should be in.

The code in Figure 10 shows a simple example of how to generate 6 routes and 6 destinations within a zone of radius 50m (obviously not realistic, just for the purpose of illustration). Figure 11 shows the result of running this.



```
seed = 2019
routes_gen = RoutesGenerator(seed=seed)
routes_gen.generate(n_destinations=6,
                    n_routes=6,
                    zone_radius=50,
                    origin=(0., 0.),
                    sub_routes_ratio=0.0)
routes_gen.plot_routes(annotate=True, save_path='example_route_generator.png')
routes, destinations = routes_gen.process_get_dataset()
print("Routes: \n{}\n".format(routes))
print("Destinations: \n{}\n".format(destinations))
```

Figure 10 Data Generation Code

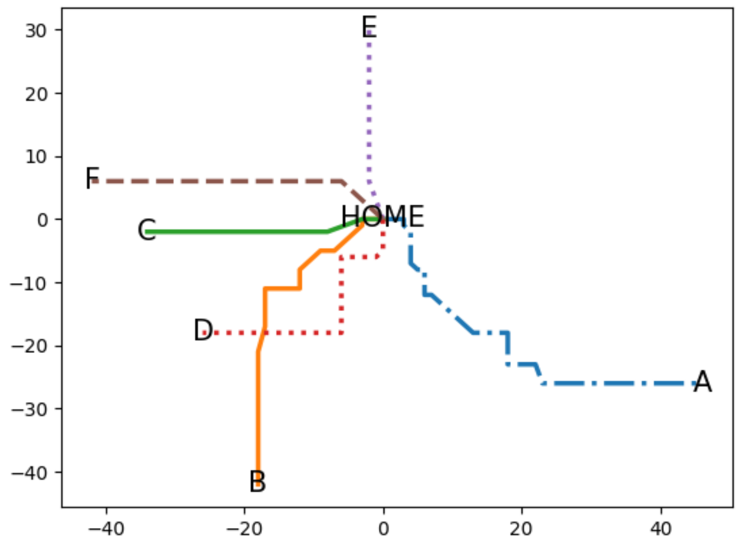


Figure 11 Sample Routes

Data Pre-processor

This module applies the required pre-processing on the Routes Dataset, namely

1. **Padding**: pads the sequence of routes with zeros to make them all of same length.
2. **Features Scaling**: since routes are dynamic and vary in their length and coordinates, sequence scaling might be necessary to bring high numbers within a consistent numeric range. Available scaling's are normalization, standardization, robust scaling, maximum absolute value scaling.
3. **Label Encoding**: encodes labels with value between 0 and n_classes-1 in order to convert non-numeric values to numeric ones.
4. **OneHot Encoding**: encodes categorical integer features as a one-hot numeric array to suit for the categorical cross-entropy loss function.
5. **Dataset Splitting**: split arrays or matrices into random train and test subsets.

ML Model

This module defines the logic of the model and builds it. As well as facilitating the following functions:

1. **Train**: trains the model fully on a given dataset and evaluates on a ratio of observations taken from the end of the data as a cross-validation split.
2. **Partial Train**: partially trains the model on a given dataset where routes are obtained separately (i.e. online training).
3. **Predict**: predicts the class probability of given route sequence and returns it as a vector.
4. **Reconstruct**: reconstructs the model architecture in case a new destination is added or an existing one removed. It transfers parts of the original trained model to the new model.

Performance Analysis

This module provides the most important dynamic evaluation tools for multi-class routes classification, namely:

1. **Plot Training History**: plots the training history e.g. training accuracy, training loss, validation accuracy, validation loss.
2. **Plot ROC Curve and AUC Scores**: compares the sensitivity vs specificity i.e. compares the true positive rate and false positive rate.
3. **Plot Confusion Matrix**: Summarizes the performance of the model for a better idea of what the model is getting right and what types of errors it is making.
4. **Create Classification Report**: builds a text report showing the main classification metrics e.g. accuracy, precision, recall, F1-Score, support, macro average, and weighted average.

Routes Utilities

A utilities module providing a variety of helper functions including GridSearchCV and Early Stopping Callback

1. **GridSearchCV**: performs exhaustive search over specified parameter values for the routes model on one, very few, or a complete set of routes. It returns a dictionary of the best model and its information along with best parameters (based on the overall average performance).
2. **EarlyStopping**: a callback that terminates training when one or all metrics reach the specified baseline(s). It is very useful to avoid overfitting and underfitting, as well as terminating training once the model has learned enough. So, no more redundant training epochs or divergence after convergence.

Model Logic and Architecture

Figure 12 Model Architecture shows the overall model architecture and its logic. The convolution layer here is used for extracting local sub-sequences from the route main sequence and to identify local patterns within the window of convolution i.e. augmenting sequence with its subsequence's based on the number of filters specified.

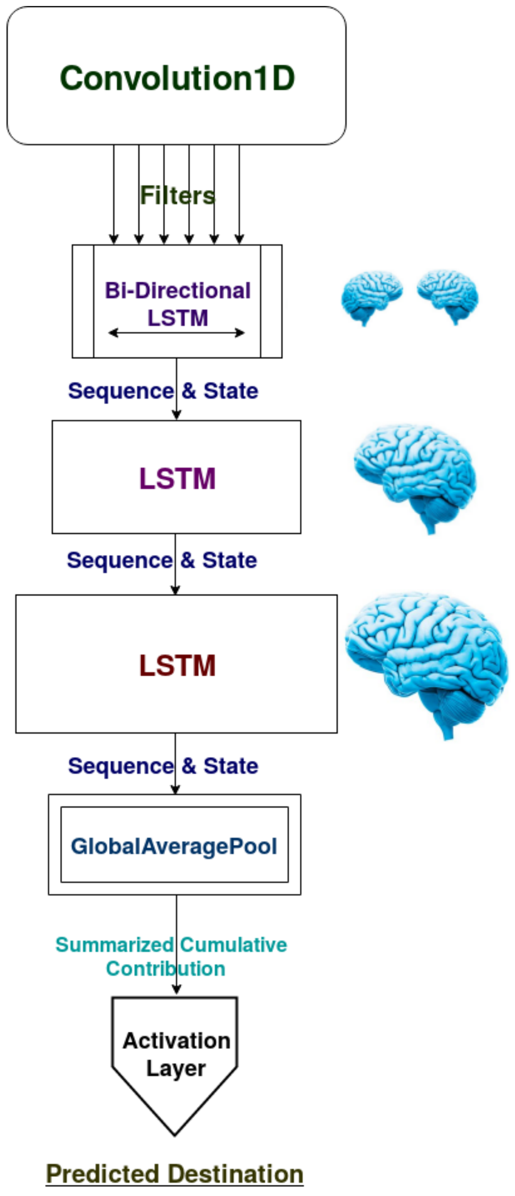


Figure 12 Model Architecture

Stacking LSTM hidden layers makes the model deeper, more accurately earning the description as a deep learning technique. Since each LSTM is defined mainly by its number of units, which specifies the capacity of the LSTM memory, the number of units plays a critical role in LSTM learning quality. If the number of units is too large, convergence will be faster, but it will be prone to overfitting. On the other hand, if the number of units is too small, it takes longer to converge and will be prone to underfitting. Therefore, we build 3 LSTMs with 3 different memory capacities so each one can learn differently and cope with the different sequence lengths.

The first LSTM has lesser memory capacity, so it can perform well on small sub-sequences. However, it is bi-directional so it can see the past and the future of the sequence steps context in the route for better classification. The second LSTM has bigger memory and the third has the biggest among them all so it can remember long routes.

Next we have a Global Average Pool (GAP) to summarize the cumulative contribution effects of the entire sequence so it allows to have sequences of any length, it does that by taking an average of every incoming feature map. This GAP acts as a structural regularizer, which natively prevents overfitting for the overall structure.

Finally, there is the Activation Layer which is responsible for transforming the summed weighted input from previous layer into an output (i.e. prediction). If the number of destinations is 1, a sigmoid and a binary cross-entropy are selected automatically as an activation function and loss function respectively. Whereas if the number of destinations is greater than 1 (as would be expected), a softmax and a categorical cross-entropy are selected.

Experiments

Since the trials were greatly restricted due to the Covid-19 pandemic the route prediction had to be tested using synthetic datasets as described previously. Even though this was far from ideal we did strive to test using very challenging datasets where the routes highly overlap as illustrated in Figure 13. There are 30 destinations and 70 main routes which extremely overlap.

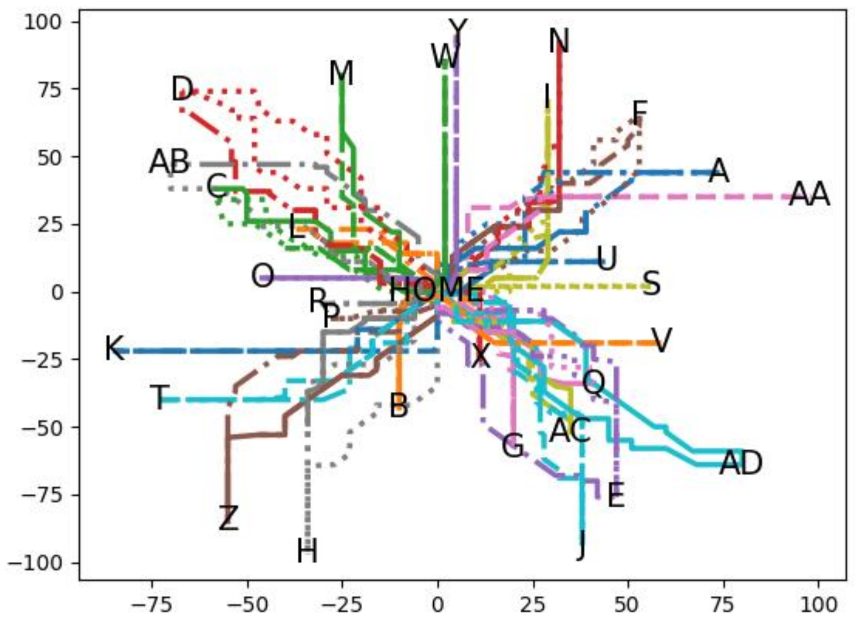


Figure 13 Challenging Routes Dataset

30 sub-sequences were created, where each sequence length is only 50% of its corresponding main route, for validating the system. In order to search for the best model, we searched through the following parameter grid

```
seed = 2019
n_dest = 30
params_grid = {'units': [25, 50, 100],
               'n_dest': [n_dest],
               'n_features': [2],
               'optimizer': [Adadelta(lr=0.001),
                             Adamax(lr=0.001),
                             Nadam(lr=0.001),
                             Adam(lr=0.001),
                             RMSprop(lr=0.001)],
               'n_filters': [32, 64, 96],
               'kernel_size': [5, 10, 20],
               'recurrent_dropout': [0.0, 0.1],
               'seed': [seed],
               'epoch': [30],
               'shuffle': [False]}
```

Figure 14 Parameter Grid

After running the GridSearchCV on the above parameters grid using an early stopping configuration, the following best set of parameters were found (non-important parameters omitted)

- units: 50
- optimizer: Nadam
- n_filters: 64
- kernel_size: 10
- recurrent_dropout: 0.0

The reason we performed a GridSearchCV is because the best optimizer, LSTM memory capacity, and number of filters in CNN etc. differ in different scenarios and on different datasets. Although the above parameters were the best in this situation (e.g. the best optimizer found to be Nadam), repeating the experiment on different datasets might well yield quite different results because there is no one solution to every situation.

Below are included some of the results of running this experiment.

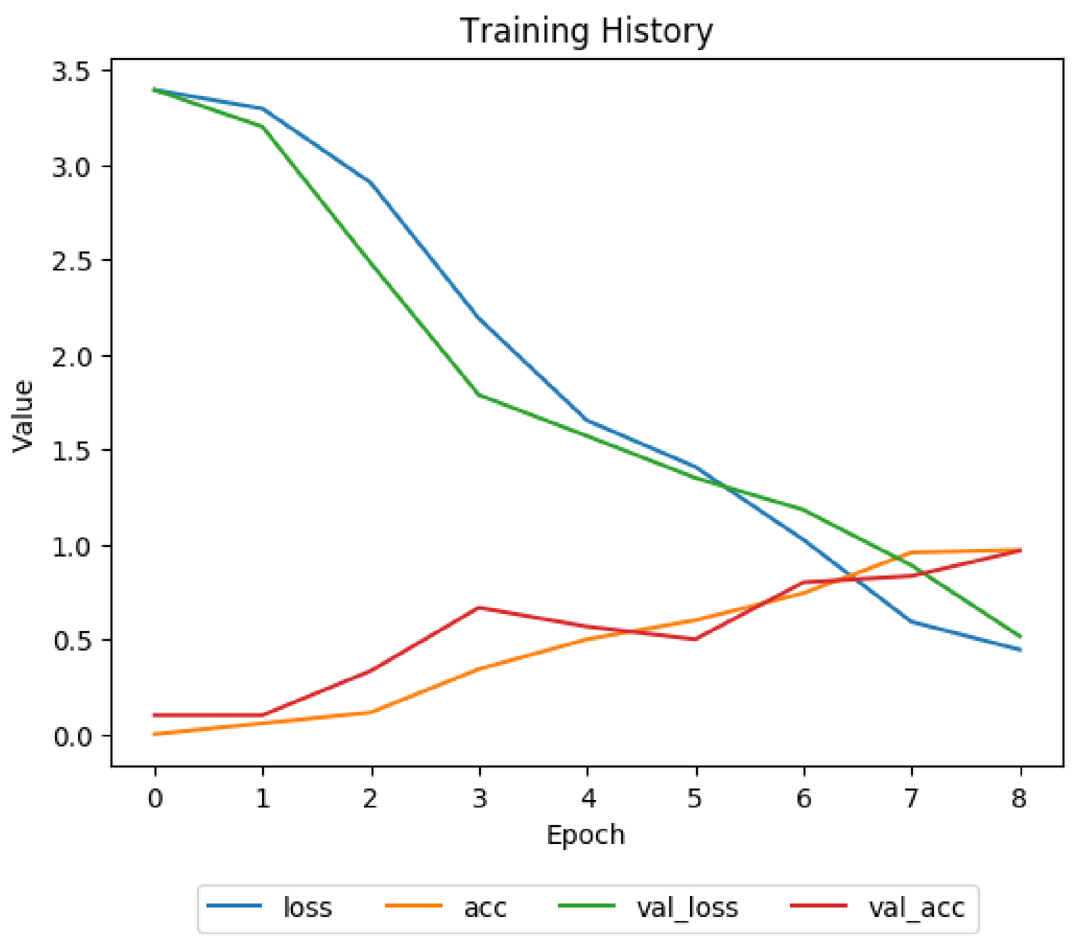


Figure 15 Training History

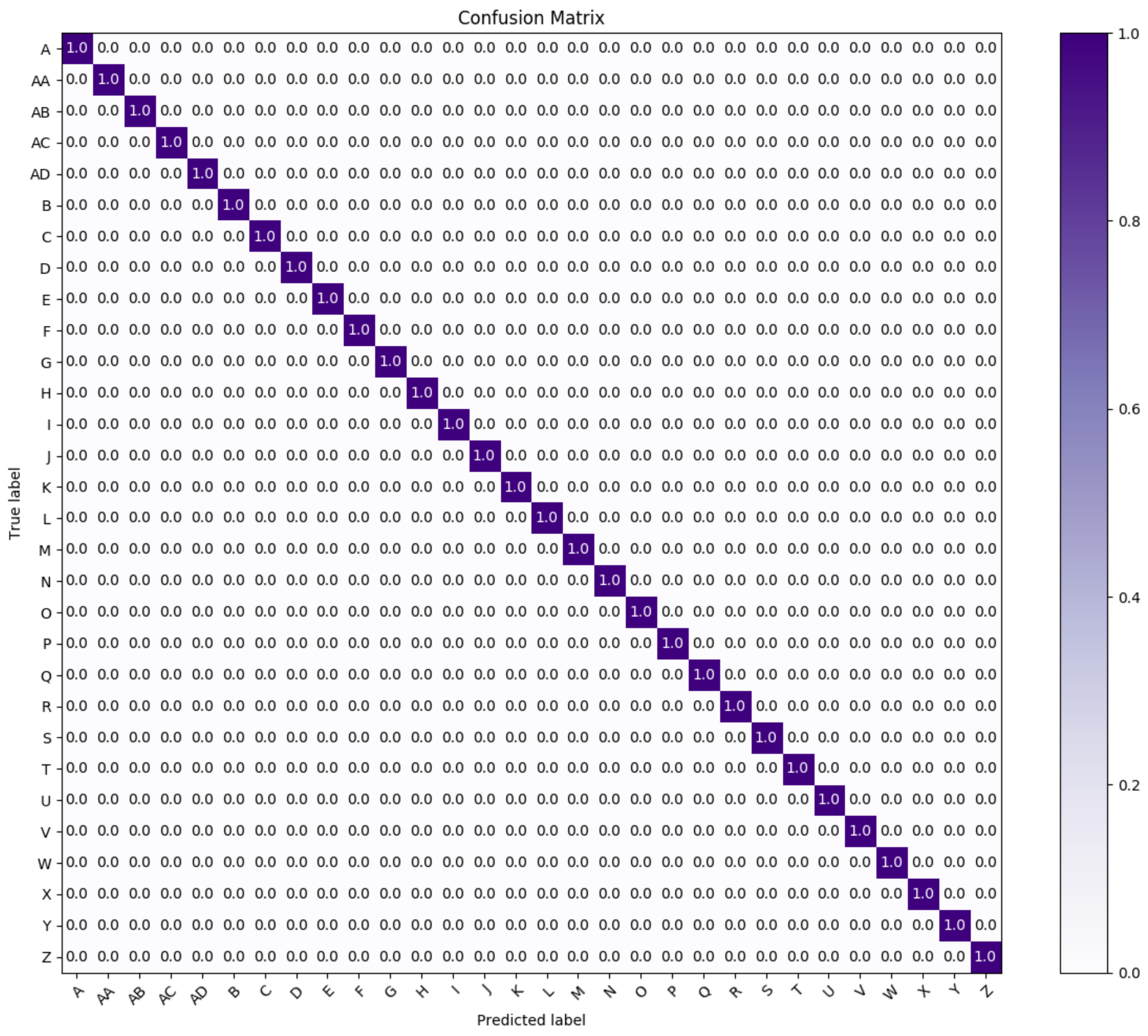


Figure 16 Confusion Matrix

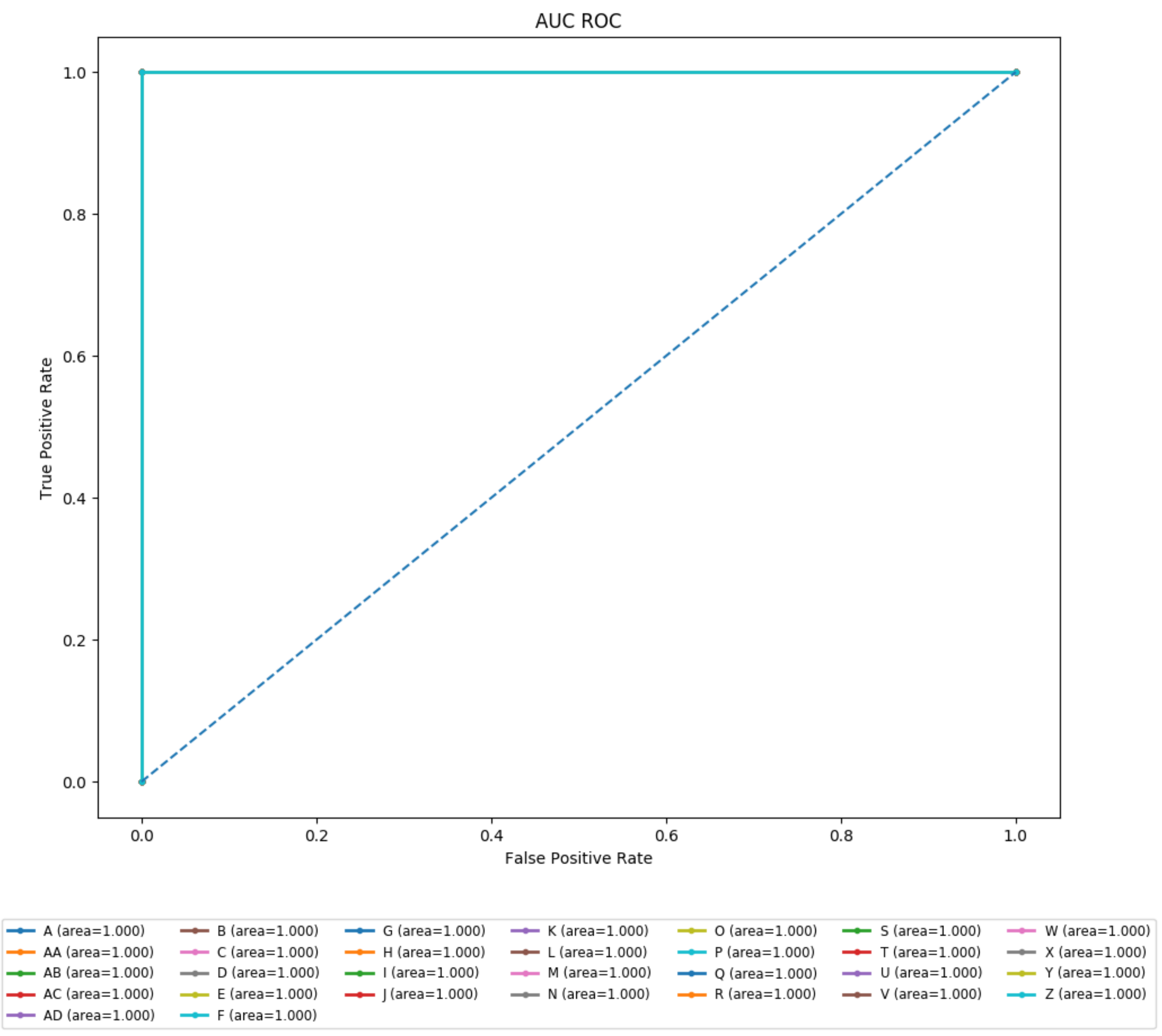
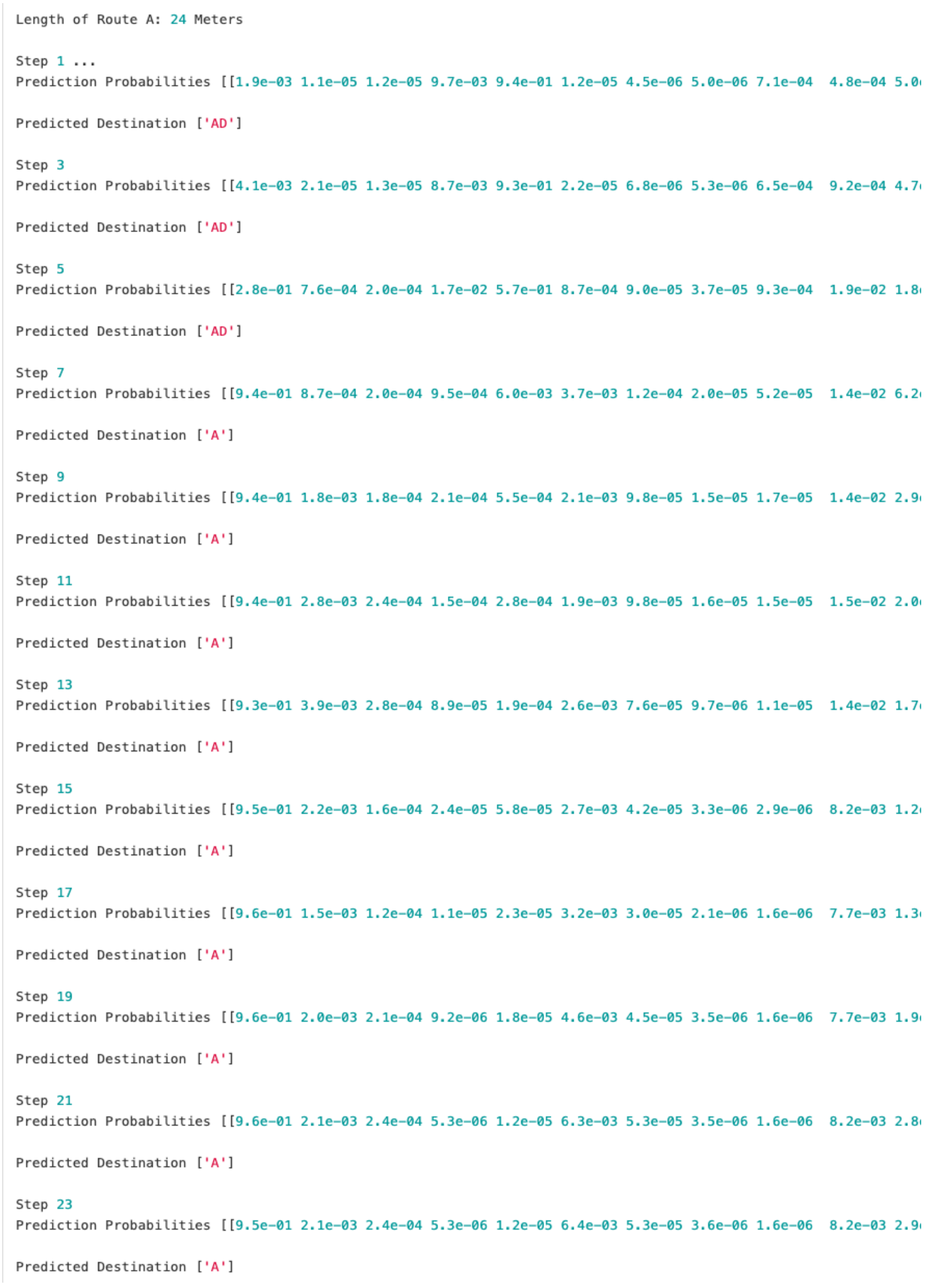


Figure 17 ROC AUC Plot

Dynamic Predictions

An important feature of such an algorithm is its ability to quickly determine the likely destination a user is going towards. To test this, we picked arbitrary routes and attempted to predict the sub-sequences along the way, starting from the original point towards the destination with 2 steps taken before the next prediction. Essentially, we are simulating a PwD moving towards some destination (destination 'A' in the example below) and checking in real-time how early the AI model can guess the correct destination as the patient moves, taking into consideration the complexity of the model and the overlapping of routes.



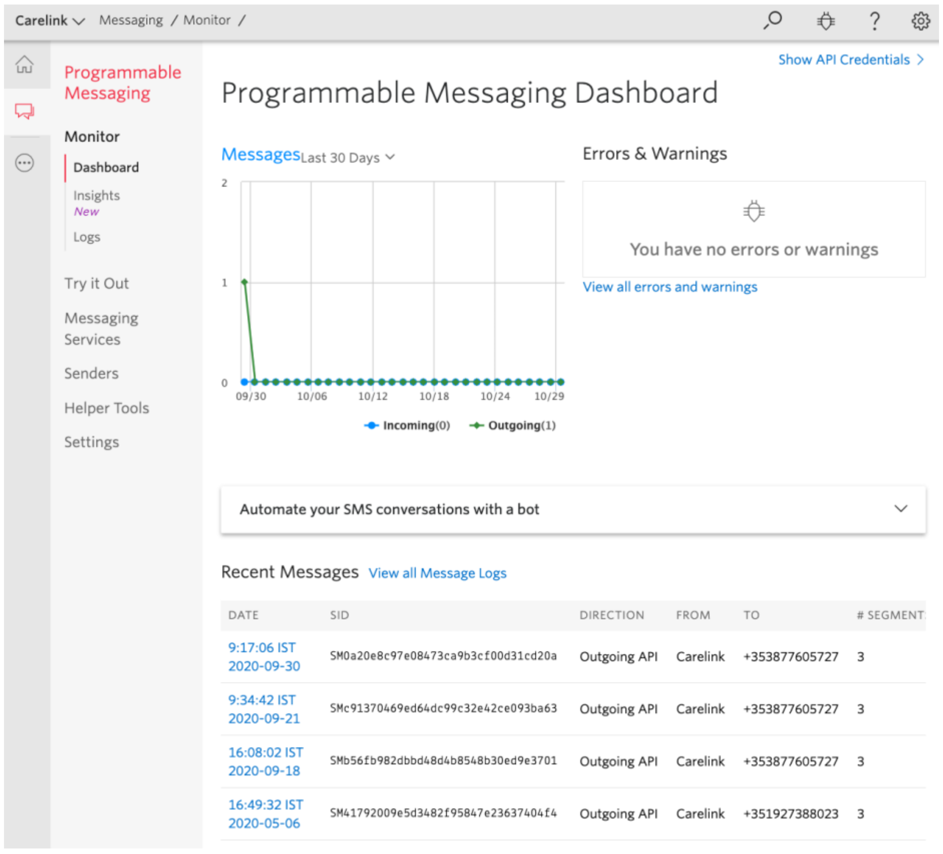
```
Length of Route A: 24 Meters

Step 1 ...
Prediction Probabilities [[1.9e-03 1.1e-05 1.2e-05 9.7e-03 9.4e-01 1.2e-05 4.5e-06 5.0e-06 7.1e-04  4.8e-04 5.0

Predicted Destination ['AD']

Step 3
Prediction Probabilities [[4.1e-03 2.1e-05 1.3e-05 8.7e-03 9.3e-01 2.2e-05 6.8e-06 5.3e-06 6.5e-04  9.2e-04 4.7

Predicted Destination ['AD']

Step 5
Prediction Probabilities [[2.8e-01 7.6e-04 2.0e-04 1.7e-02 5.7e-01 8.7e-04 9.0e-05 3.7e-05 9.3e-04  1.9e-02 1.8

Predicted Destination ['AD']

Step 7
Prediction Probabilities [[9.4e-01 8.7e-04 2.0e-04 9.5e-04 6.0e-03 3.7e-03 1.2e-04 2.0e-05 5.2e-05  1.4e-02 6.2

Predicted Destination ['A']

Step 9
Prediction Probabilities [[9.4e-01 1.8e-03 1.8e-04 2.1e-04 5.5e-04 2.1e-03 9.8e-05 1.5e-05 1.7e-05  1.4e-02 2.9

Predicted Destination ['A']

Step 11
Prediction Probabilities [[9.4e-01 2.8e-03 2.4e-04 1.5e-04 2.8e-04 1.9e-03 9.8e-05 1.6e-05 1.5e-05  1.5e-02 2.0

Predicted Destination ['A']

Step 13
Prediction Probabilities [[9.3e-01 3.9e-03 2.8e-04 8.9e-05 1.9e-04 2.6e-03 7.6e-05 9.7e-06 1.1e-05  1.4e-02 1.7

Predicted Destination ['A']

Step 15
Prediction Probabilities [[9.5e-01 2.2e-03 1.6e-04 2.4e-05 5.8e-05 2.7e-03 4.2e-05 3.3e-06 2.9e-06  8.2e-03 1.2

Predicted Destination ['A']

Step 17
Prediction Probabilities [[9.6e-01 1.5e-03 1.2e-04 1.1e-05 2.3e-05 3.2e-03 3.0e-05 2.1e-06 1.6e-06  7.7e-03 1.3

Predicted Destination ['A']

Step 19
Prediction Probabilities [[9.6e-01 2.0e-03 2.1e-04 9.2e-06 1.8e-05 4.6e-03 4.5e-05 3.5e-06 1.6e-06  7.7e-03 1.9

Predicted Destination ['A']

Step 21
Prediction Probabilities [[9.6e-01 2.1e-03 2.4e-04 5.3e-06 1.2e-05 6.3e-03 5.3e-05 3.5e-06 1.6e-06  8.2e-03 2.8

Predicted Destination ['A']

Step 23
Prediction Probabilities [[9.5e-01 2.1e-03 2.4e-04 5.3e-06 1.2e-05 6.4e-03 5.3e-05 3.6e-06 1.6e-06  8.2e-03 2.9

Predicted Destination ['A']
```

Figure 18 Dynamic Predictions

As shown in the preceding example the dynamic prediction works very well. The model could predict the correct destination ('A') after only 7 steps[4], 29% of the overall route distance.

3.3.3.5 *Alerting Service*

The Alerting Service is quite straightforward. It listens to the 'alerts' topic on MQTT and sends an alert if requested. We investigated several different techniques for notifications from SMS, to Web Push notifications, to native app notifications. We believe that the most useful (in terms of impact, urgency, and adoption) are SMS notifications. We trialled both Amazon Simple Notification Service (https://aws.amazon.com/sns/) and Twilio (https://www.twilio.com/). For ease of use we proceeded with Twilio, which means that with just a few lines of code we can send texts. The SMS sent by Twilio includes a link that brings the carer directly to a map highlighting the location of the PW.



Programmable Messaging Dashboard

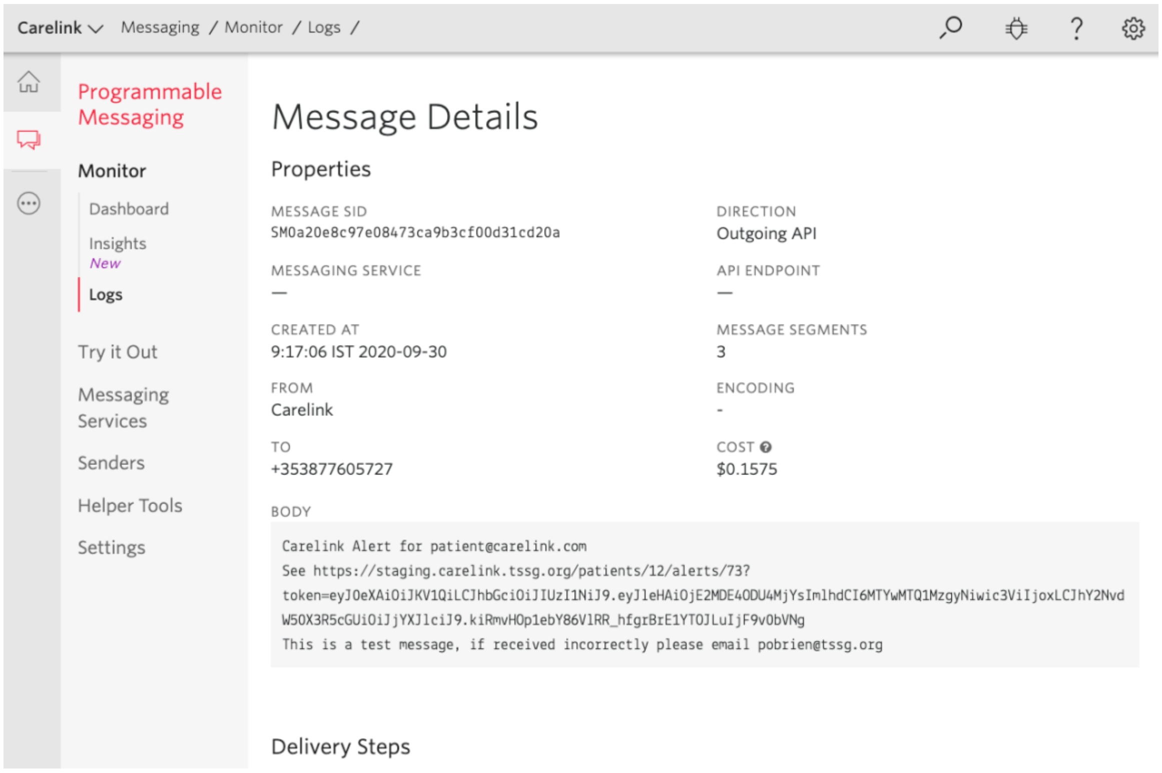
You have no errors or warnings

Recent Messages

DATE	SID	DIRECTION	FROM	TO	# SEGMENT
9:17:06 IST 2020-09-30	SM0a20e8c97e08473ca9b3cf00d31cd20a	Outgoing API	Carelink	+353877605727	3
9:34:42 IST 2020-09-21	SMc91370469ed64dc99c32e42ce093ba63	Outgoing API	Carelink	+353877605727	3
16:08:02 IST 2020-09-18	SMb56fb982dbbd48d4b8548b30ed9e3701	Outgoing API	Carelink	+353877605727	3
16:49:32 IST 2020-05-06	SM41792009e5d3482f95847e23637404f4	Outgoing API	Carelink	+351927388023	3

Figure 19 Twilio Dashboard

[4] 'steps' represent coordinate updates of a route, not physical steps taken



Message Details

Properties

MESSAGE SID
SM0a20e8c97e08473ca9b3cf00d31cd20a

DIRECTION
Outgoing API

MESSAGING SERVICE
—

API ENDPOINT
—

CREATED AT
9:17:06 IST 2020-09-30

MESSAGE SEGMENTS
3

FROM
Carelink

ENCODING
-

TO
+353877605727

COST
$0.1575

BODY

```
Carelink Alert for patient@carelink.com
See https://staging.carelink.tssg.org/patients/12/alerts/73?
token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE2MDE4ODU4MjYsImlhdCI6MTYwMTQ1MzgyNiwic3ViIjoxLCJhY2Nvd
W50X3R5cGUiOiJjYXJlciJ9.kiRmvHOp1ebY86VlRR_hfgrBrE1YTOJLuIjF9v0bVNg
This is a test message, if received incorrectly please email pobrien@tssg.org
```

Delivery Steps

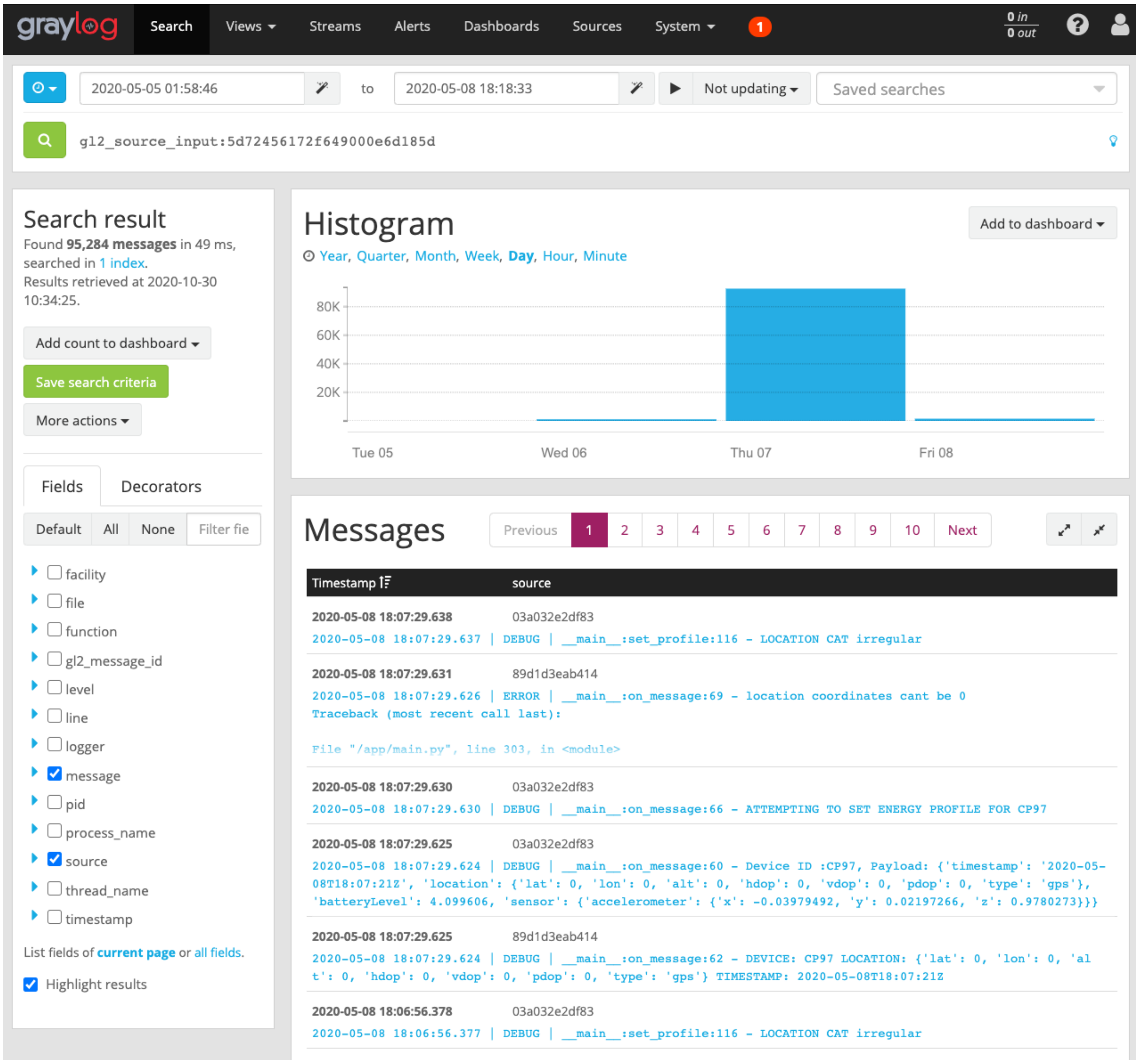
Figure 20 SMS Alert Template

3.3.3.6 Proxy Service

We began by using Nginx as a proxy service that acts as a reverse proxy to any of the other microservice APIs. We then switched to Traefik which is a leading modern reverse proxy and load balancer that makes deploying microservices easy. Traefik intercepts and routes every incoming request to the corresponding backend services. In terms of security benefits its supports SSL termination and we use it with Let's Encrypt for automatic certificate generation.

3.3.3.7 Logging Service

An important cross cutting concern for any system is logging. We had originally looked at implementing the Riemann monitoring framework as discussed in D4.2, but on further consideration we decided it was better to consider logging and monitoring as two distinct services. Sentry was used as a monitoring platform and for logging we opted for Graylog. This seamlessly collects, enhances, stores, and analyses log data from the various Carelink services. It is particularly useful as a debugging aid for device/platform communications.



```
gl2_source_input:5d72456172f649000e6d185d
```

Search result

Found **95,284 messages** in 49 ms, searched in 1 index.
Results retrieved at 2020-10-30 10:34:25.

- ☐ facility
- ☐ file
- ☐ function
- ☐ gl2_message_id
- ☐ level
- ☐ line
- ☐ logger
- ☑ message
- ☐ pid
- ☐ process_name
- ☑ source
- ☐ thread_name
- ☐ timestamp

List fields of **current page** or all fields.

☑ Highlight results

Histogram

Messages

```
Timestamp               source
2020-05-08 18:07:29.638 03a032e2df83
2020-05-08 18:07:29.637 | DEBUG | __main__:set_profile:116 - LOCATION CAT irregular

2020-05-08 18:07:29.631 89d1d3eab414
2020-05-08 18:07:29.626 | ERROR | __main__:on_message:69 - location coordinates cant be 0
Traceback (most recent call last):

File "/app/main.py", line 303, in <module>

2020-05-08 18:07:29.630 03a032e2df83
2020-05-08 18:07:29.630 | DEBUG | __main__:on_message:66 - ATTEMPTING TO SET ENERGY PROFILE FOR CP97

2020-05-08 18:07:29.625 03a032e2df83
2020-05-08 18:07:29.624 | DEBUG | __main__:on_message:60 - Device ID :CP97, Payload: {'timestamp': '2020-05-
08T18:07:21Z', 'location': {'lat': 0, 'lon': 0, 'alt': 0, 'hdop': 0, 'vdop': 0, 'pdop': 0, 'type': 'gps'},
'batteryLevel': 4.099606, 'sensor': {'accelerometer': {'x': -0.03979492, 'y': 0.02197266, 'z': 0.9780273}}}

2020-05-08 18:07:29.625 89d1d3eab414
2020-05-08 18:07:29.624 | DEBUG | __main__:on_message:62 - DEVICE: CP97 LOCATION: {'lat': 0, 'lon': 0, 'al
t': 0, 'hdop': 0, 'vdop': 0, 'pdop': 0, 'type': 'gps'} TIMESTAMP: 2020-05-08T18:07:21Z

2020-05-08 18:06:56.378 03a032e2df83
2020-05-08 18:06:56.377 | DEBUG | __main__:set_profile:116 - LOCATION CAT irregular
```

Figure 21 Graylog Dashboard

3.3.3.8 Supporting Services

In addition to the services described in the previous sections there were a number of additional services. These used the official Docker images and so are just listed here for completeness.

- Eclipse Mosquitto as the MQTT broker
- Swagger UI to host the API documentation
- Postgres as the database, with the PostGIS spatial database extender
- Redis as a cache
- Node-RED to enable communication using LoRa as well as using wifi geo-location services to assist with tracking. It will be used in the backend to do data processing of the packets received by The Things Network (TTN). The devices will communicate through LoRa with a gateway. This gateway will forward the packets to TTN, which will then communicate by MQTT to Node-RED, which in turn will communicate by MQTT with the Carelink platform. Node-RED is also used to process the 'status' messages (that are published in the MQTT broker, with a field containing the Wi-Fi access points that are near the device), which will make use of 3 Wi-Fi location API's to determine an "assisted" location, complementary to the GPS. This will enable the Pycom devices to have one more alternative to the GPS, in case of failure.

3.4 Communications

3.5 Privacy and Security

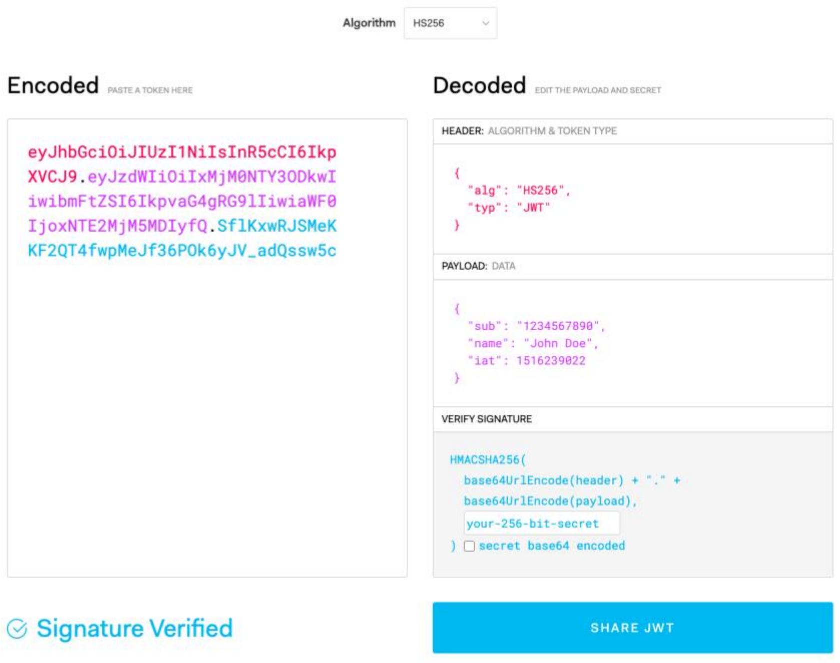
Main points regarding privacy are detailed in D2.3. All communication between the web service and the platform is done over encrypted, authenticated, and authorised channels.

The decoupled nature of the services described in the previous section bring many benefits, but they also introduce complexity chief among which is how to secure the services without introducing dependencies and breaking the decoupling.

The first line of security is SSL. All traffic to/from the Carelink application will be encrypted – no information between the application and client should be sent in the clear. This mitigates against man in the middle attacks and prevents sensitive information from being sniffed.

Access to the Carelink platform will be further restricted to authorised users. Users will have to authenticate themselves to the system in a Single Sign On (SSO) mechanism which will require that the user be registered (this happens through the User Service API).

Upon successful authentication, a user session is created which will be maintained until such time as they sign out, or the session expires. The session will maintain a user context which will be attached to every request from the client. Carelink will use JSON Web Tokens (JWT) as specified in RFC7519 to secure inter service requests and provided the user context to services. The token will encrypt and sign the user context using an RSA key pair. On receipt, a service can validate the source of the request via the public key and thus have access to the context. Furthermore, the token can be passed on verbatim to any secondary service that needs to be called to complete the request. These tokens will have a limited lifetime, which will mitigate against misuse.



Algorithm HS256

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp
XVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwI
iwibmFtZSI6IkpvaG4gRG9lIiwiaWF0
IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeK
KF2QT4fwpMeJf36POk6yJV_adQssw5c
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Figure 22 JWT Example

4 Data and API Design

4.1 Data Model

The Carelink Ontology model represents the conditions, rules and assertions, required for the interoperability, configuration and management of the user's, hardware feature's and device's profiles.

Depicted in **Figure 23**, is a simplified diagram of the ontology model, where the classes are represented in yellow, their respective datatype properties are represented in green, and the object properties are represented in blue, which characterize the relationships and rules of the classes. The cardinality of each relation, either of the datatype or object properties, is represented as mathematical inequalities over the arrows.

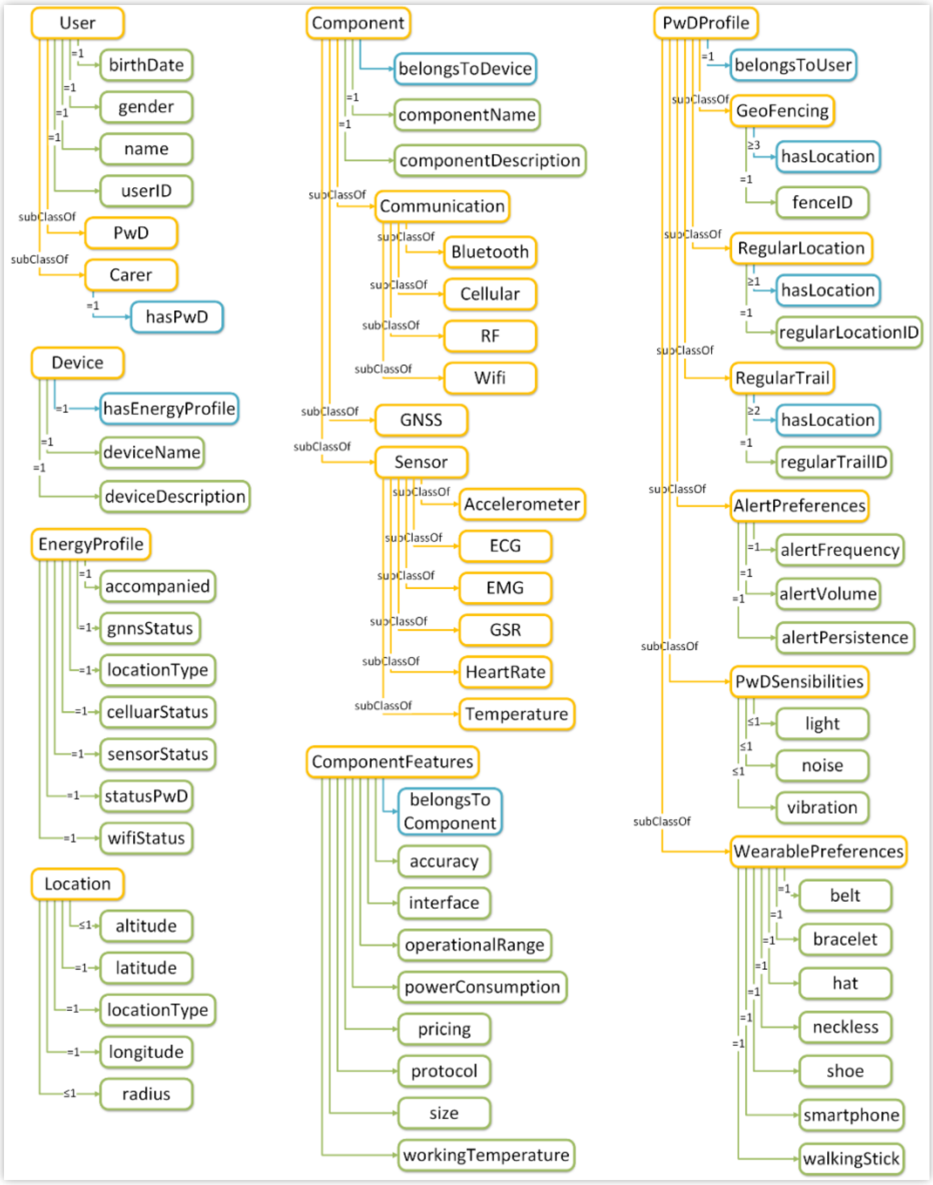


Figure 23 Carelink Ontology Model Diagram

There are seven proposed primary classes, namely *User, EnergyProfile, Location, Device, PwDProfile, Component and ComponentFeatures.*

The *User* class represents the identification of the target audience of the Carelink system, i.e. the *PwD* and the *Carer*.

The *PwDProfile* represents the aggregation of the *PwD* preferences, his usual location history and settings, and is used to tune the Carelink solution to each individual need and requirement. These include *GeoFencing, RegularLocation, RegularTrail,* which are supported by the *Location* class, and the *AlertPreferences, PwDSensibilities and WearablePreferences*, which tailor the device application to the PwD needs*.*

The *Device* class represents the physical solution that will accompany the *PwD*, which is comprised of different components and features, each represented in the *Component* and *ComponentFeatures* classes. A *Component* type can be *Communication* (either *Cellular, Wifi, Bluetooth* or *RF*)*, GNSS* or *Sensor* (either an *Accelerometer, ECG, EMG, GSR, HeartRate* and/or *Temperature* sensors).

The *EnergyProfile* class represents the conditions and status of the device for the implementation of a smart adaptable energy management service, that can enable or disabling each *Component* depending on whether or not it is required.

5 Supporting Infrastructure

5.1 Data Centre Overview

The TSSG house and operate their own Data Centre facility to support over 50 concurrently active ICT research projects through the provisioning of Internet services, Cloud Computing resources, and project bespoke testbeds (such as Unified Communications, SDN/NFV, Internet of Things, etc.) and research equipment. The TSSG Data Centre also houses research infrastructure for other research groups and centres, such as ICHEC's Super Computer 'Fionn' and CONNECT's Pervasive Nation Testbed.

Figure 24 Data Centre Interior

The TSSG Data Centre currently supports over 160 physical servers, providing more than 1,000 CPU cores for processing and 400 virtual servers for cloud computing, including a selection of NVIDIA Tesla K20 GPUs for AI and Deep Learning. In addition, there is over ½ PB (that 512 TB) of Data Storage, and ~3,000 network ports. All of which provide a high-level of interconnectivity and flexibility for TSSG's research projects.

A large OpenStack cluster is hosted in this data centre, providing infrastructure-as-a-service capabilities. All of this is available to the Carelink project. We are currently using the virtual servers to host the platform, and leveraging the GPUs to train some of the Carelink algorithms.

6 CONCLUSIONS

The Carelink solution, comprising software and hardware, has been carefully implemented with the clear goal of providing a robust, energy-efficient means of tracking the location of a PwD, and alerting their carer in the event of a wandering episode. The choice of devices and software tools, from the geospatial database to the lightweight MQTT messaging has always been driven by this goal. As well as the whole solution works together, crucially, it was designed with the tenets of composability and extensibility, so that in all endeavours to bring this to market those components adding the most value in any potential exploitation opportunity can be leveraged.