Generating Safe Routes for Pedestrians: Avoiding Crime and Unsafe Roads

A dissertation submitted in partial fulfilment of the University of Greenwich undergraduate degree programme:
BSc (Hons) Computer Science

Lucas Gomes - Banner ID: 001051104
Supervisor: Jia Wang

July 2022

Abstract

This article aims to provide methods for planning a safe route for pedestrians. This was achieved by creating a cost algorithm that takes in consideration crime density on streets, and road data provided by OpenStreetMaps. The algorithm effectively creates different and safer routes than popular navigations software such as Google Maps where it only focus on the fastest route. To prove the usability of the algorithm, an interactive Web Application was also created.

1 Introduction

According to the Telephone-operated Crime Survey for England and Wales (TCSEW), adults aged 18 years and over experienced 12.8 million offences in the year ending December 2021. Which, compared to the year 2020, is an 18% increase in total crime, with a 54% increase in fraud and computer misuse offences and a 15% decrease in theft offences (Office for National Statistics 2022).

- There were also crimes recorded by the police that indicated a 4% decrease in the number of offences involving knives or sharp instruments (knife-enabled crime) to 46,950 offences in the year ending December 2021, from 49,152 in the year ending December 2020. While it decreased from year ending 2020 to 2021, it increased 4.9% from the United Kingdom's(UK) pre-lockdown(Institute for Government 2021)(Johnson 2020a)(Johnson 2020b) year ending December 2019(Office for National Statistics 2020) with 44771 offences to the year ending December 2021 with 46950 offences(Office for National Statistics 2022).

- A 5% decrease in offences involving firearms compared to the year 2020; While there was growth in last nine months of 2021, the levels remained lower than the pre-coronavirus year ending on March 2019 (Office for National Statistics 2022).

- There were 691 homicides recorded in the year ending December 2021, indicating an increase of 14% over the 604 homicides crimes that happened in the year ending December 2020.(Office for National Statistics 2022).

For the purposes of assisting on the decrease of criminal offences, this article introduces a cost algorithm for path finding algorithms such as Dijkstra's in a street network environment to determine the safest pedestrian route, to avoid the mentioned offences. Since most navigation software such as Google Maps(Google 2022b) only generate the quickest route and not the safest, the article also presents a Web Application that allows the user to obtain safe routes from a desired start location and destination in a restricted area containing Stratford and East Ham.

2 Background

The author did not find many research articles about the specific topic of safe pedestrian route planning, so the research was done using the following keywords: 'Crime prediction', 'route planning', 'path planning crime' and 'safe route planning'.

Utamima and Djunaidy (2017) found a lack of software that could calculate a safe path as a vehicle passenger. So, their research resulted on web-app called "Be Safe Travel" that creates routes that might be considered safe by avoiding crime points. It can also exhibit the most dangerous route. For the tools, it used the MySQL database to store the crime points. To display the routes, it used the Google Maps API.

Aziz and Yusoff (2020) introduced an algorithm to obtain traffic density using a sample data from Automated Vehicle Locator Systems (AVLS) and crime data. With the assistance of GIS, the research accomplished a visualization of crime clusters, patrolling patterns, and crime locations that are not being patrolled.

Klochovka et al. (2017) proposed an enhanced representation of the pedestrian network that provides benefits for pedestrian route planning. The representation was done by using pavements, crossings between them, travel time derived from distance and average walking speed. Additional delays were calculated using probabilistic methods for signal crossings. The authors utilized the PostgreSQL extension PGRouting to compute the

routes within the study area using the A* shortest path algorithm (Hart, Nilsson, and Raphael 1968).

Yang (2019) experimented geospatial technologies to support efforts of community organizations and of the Chicago Gun Violence Research Collaboration to reduce gun-related crimes. This research used data of crime incidents in the city of Chicago to conduct spatial analyses and gather statistics based on Kernel density, optimized hotspot analysis, and emerging hotspot analysis. A mobile app was created that contained interactive maps, allowing the public to view the spatial patterns of gun-related crimes and related outputs. It also enabled the users to control the maps, providing potential ancillary data for the researchers and community residents.The author concluded that spatial information generated and displayed in a GIS environment can make advanced communication easier between community residents and research groups and urban policy makers. Additionally, assisting all stakeholders on understanding community needs and resources.

Yao et al. (2017) created a Multi-Objective Hyper-Heuristic(MOHH)(Burke et al. 2013) algorithm that uses reinforcement learning. It uses data from the city of New York, which includes historical crime data, taxi trip data, housing rent price, population, position of police stations and other Point Of Interests(POI's). According to the authors, it is 1.4 times faster than the Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al. 2002) and 34 times faster than the multi-objective optimization algorithm.

Crime forecasting can be useful for determining a route's safety:

- Islam and Raza (2020) achieved 80% of accuracy when predicting the number of crimes across the years due to their model based on Auto Regression Integrated Moving Average(ARIMA)(Box and Pierce 1970).
- Khalid, Shoaib, Qian, Rui, A. I. Bari, et al. (2018) used Spatial Analysis along Networks(SANET) for computing the Network Kernel Density Estimation (NetKDE) which the results of it were passed as inputs to the Getis-Ord GI* statistics. which their output resulted on clusters. From those clusters, it was observed the following:

  - Bike theft incidents tend to happen on the busy road around the public facilities and in commercial areas of the city while the car theft incidents are located in busy markets and unattended parking lots.

  - The incidents of robbery are clustered in the residential and industrial areas while the snatching incidents are clustered in the residential and commercial parts of the city.

3 Methods

The core idea of this project is to use PgRouting(pgRouting Community 2022) to run the Dijkstra's path finding algorithm in a street network. The cost(proportional to the danger intensity) of each street will be calculated with the proposed cost algorithm function that takes crime data and points of interest in consideration. When these elements are combined, it will result on the goal of this project, which is acquiring a safe route.

In order to create and utilize the proposed cost algorithm, the following procedures were made:

- A Geograpic Information System(GIS) to visualise the street network and routes. In this case, the QGIS software was used (QGIS contributors 2022).
- Setup a PostgresSQL(version 14.2), PostGIS(version 3.2.0), PgRouting(version 3.3.0).
- Setup a Python environment to create utilities for handling data.
- Obtain Crime data containing their location and save to a PostgresSQL Database.
- Obtain street data from OpenStreetMap(OpenStreetMap contributors 2022a) that will correlate with the crime's location.
- Create and use the cost algorithm to plan the safest route.

3.1 PostgreSQL Database, PostGIS, PgRouting

PostgreSQL

PostgreSQL is a open-source object-relational database system that uses the SQL language. It allows to store data like text, numbers, geometries and more(The PostgreSQL Global Development Group 2022c). It's the base platform that makes using PostGIS and PgRouting possible.

PostGIS

PostGIS is a spatial database extension for PostgreSQL. It adds support for geographic objects, allowing to query locations from the database (POSTGIS 2022a). For this project it enables saving crime and street locations and generate geometries based on spatial data.

PgRouting

PgRouting is an extension of PostgresSQL that has PostGIS as dependency. It provides many path-finding algorithms to generate a route(pgRouting Community 2022). In the context of this article, it allows to create a safe route.

3.2 Python environment

For this project, the programming language Python(Python Software Foundation 2022) was used to create utilities for fetching and importing data.

The pipenv(Kenneth Reitz 2022) tool was utilized to make managing virtual environments and package dependencies easier.

3.3 Crime Data

The platform data.police.uk(data.police.uk 2022b) was used to obtain crime data of London, United Kingdom, from January 2021 to December 2021. The police forces involved are the "Metropolitan Police"(Metropolitan Police 2022) and "City of London"(City of London Police 2022). The outcomes were also downloaded, but they were not used. The data comes in a Comma Seperated Values(CSV) format. Their attributes/columns are the following as described in data.police.uk (2022a):

- Crime ID: Identifier of the Crime.
- Month: The month that the crime happened. Truncated to show the year and month only.
- Reported by: The force that provided the data about the crime.
- "Longitude" and "Longitude": Crime's location coordinates.
- "LSOA Code" and "LSOA name": References to the Lower Layer Super Output Area(LSOA) that the anonymised point falls into, according to the LSOA boundaries(Office for National Statistics 2016) provided by the Office for National Statistics.
- Crime type: What category the crime falls into [1].
- Last outcome category: A reference to whichever of the outcomes associated with the crime occurred most recently. A field provided for forces to provide additional human-readable data about individual crimes. Currently, for newly added CSVs, this is always empty.
- Context: A field intended for forces that contains additional human-readable data about individual crimes. Currently, for recent CSVs, this is always has empty values.

It's important to note that the location content does not express the exact location of the crime. The coordinates are related to the position of what the data.police.uk refers to as Snap Points(data.police.uk 2022a), which are pre-defined locations that serve to protect the privacy of every individual involved in the crime. Each crime location in this dataset is attached to the closest Snap Point of its true location. Due to this method, the crime's location is accurate at most on a street level(data.police.uk 2022a), but not at post code level(Tompson et al. 2015).

Pre-Processing

The goal of crime data's pre processing is to only use relevant data from the dataset and/or format it to be easier to manipulate. Thus, crimes of the types : 'other theft' and 'other crime' were removed. Duplicated records with the same 'Crime ID' were removed.

The crime dataset is organized by months in the form of folders:

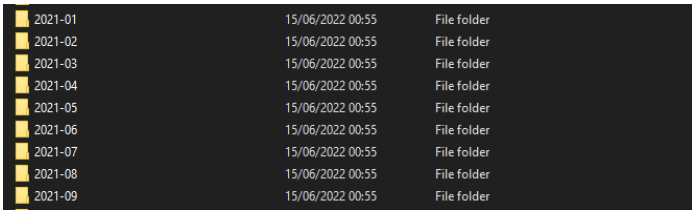

Figure 1: Folder structure of data.police.uk dataset

Inside each folder there are CSV files for each police force. For each police force there are crime files with suffix "-crime" and their respective outcomes file with the suffix "-outcomes". For this project, the crime files are used exclusively.

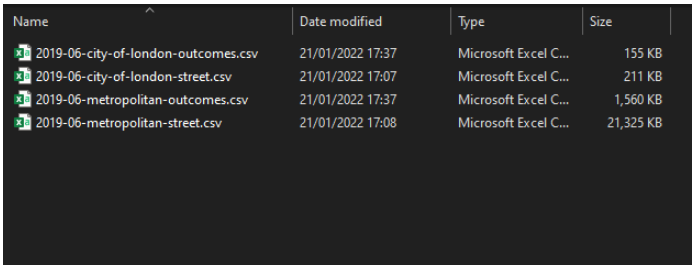

Figure 2: CSV files belonging to the month of June 2021

The first major step is to visualise and import the crime data to the PostgreSQL database. In order to facilitate this process, the crime files that occurred in 2021, from the Metropolitan and City of London police forces were merged. There are records with the same Crime ID due to their status update. To make crime counting more accurate, records with duplicated Crime ID were removed. This process was possible using Python and the packages Matplotlib(The Matplotlib Development team 2022) and Pandas(Pandas 2022).[2].

[1]The crime types can be found on the webpage: `https://www.police.uk/pu/about-police.uk-crime-data/`

[2]Script to merge and visualise is located on src/police_data.py

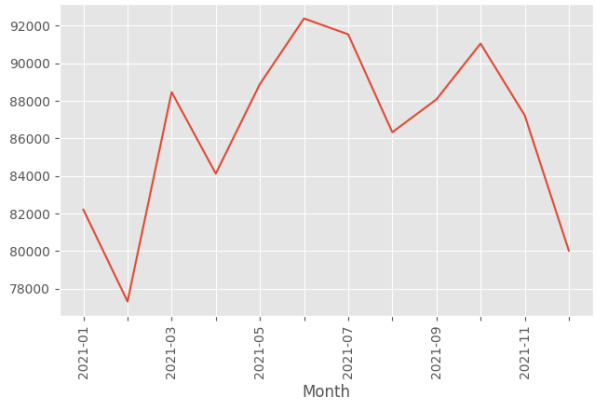

Figure 3: Number of crimes in the year of 2021.

Before importing to the database, the files related to the month of June of 2021 were also merged because it was the month containing the greatest number of crimes, that being 92390[3], which supplies enough sample quantity for testing. When importing the merged file to the database, the crimes that have the 'Crime type' attribute with the values ('other theft', 'other crime') are excluded from the importing process, resulting on 82553 records being imported out of 92390.

After importing to the database (See Fig.4 for its Schema), besides the columns that come with the original data, there are two additional columns: "crime_street_id", "created_at", "longlat_point".

- crimes_street_id - Primary Key of the table. It auto increments.
- created_at - When the record was created.
- longlat_point - It's a column of type "geography" and combines the values of "latitude" and "longitude" to create a geographic point that uses the Spatial Reference Identifier (SRID) 4326.

crimes_street		
PK	crime_street_id	int
	crime_id	varchar
	crime_month	date
	created_at	date
	reported_by	varchar
	falls_within	varchar
	latitude	varchar
	longitude	varchar
	longlat_point	geography (Point)
	location_desc	varchar
	lsoa_code	varchar
	lsoa_name	varchar
	crime_type	varchar
	last_outcome_category	varchar
	context	varchar

Figure 4: Crime Data's Schema after importing the data to the database

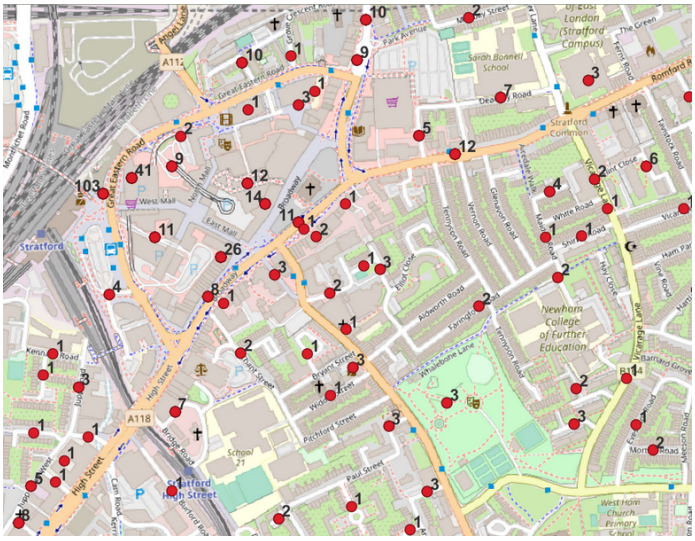

Figure 5: Sample of crime locations in Stratford(London). Due to the rime locations being placed in Snap Points, it's possible to count the crime per Snap Point, which are represented by the black numbers. (See appendix for SQL query)

[3]Without removing duplicate Crime ID's, the number of records are 96097

3.4 Geographic Data and Points of Interest (OpenStreetMap)

Geographic data is provided by the OpenStreetMap(OSM) project(OpenStreetMap contributors 2022a) and acquired via the Overpass API (Overpass API contributors 2022a). OpenStreetMap is mainly a database with data being open and populated by volunteer mappers and engineers (OpenStreetMap contributors 2022b). It contains worldwide geographic data such as streets, roads, railways, waterways and establishments like shops and offices(Overpass API contributors 2022b). The Overpass API keeps a copy of the main database up to date and provides them for search. OSM has the following data structure for its objects:

- Geometries: coordinates and references to the same coordinates. They are provided by fetching Nodes, Ways and Relations(Overpass API contributors 2022c). A Node can have a shape of a **point**, characterised by having just a longitude and latitude(OpenStreetMap contributors 2021). A Way can have a shape of a **linestring/polyine**, or **polygon**. Linestrings serve to represent linear features such as streets and rivers. Polygons serve to represent areas such as forests and buildings. (OpenStreetMap contributors 2021). Relations can be made out of previously mentioned shapes, including **multipolygons**(OpenStreetMap contributors 2021).

- Text: Information of the object that gives context to the physical world (Overpass API contributors 2022c), which can be depicted for example, in the form of tags(Overpass API contributors 2022c). Tags are attributes of the object, and each tag has a value. For example, the tag `highway` can have the value of `residential` to indicate that is a residential street.

- Metadata: Information that attributes the sources to the data(Overpass API contributors 2022c).

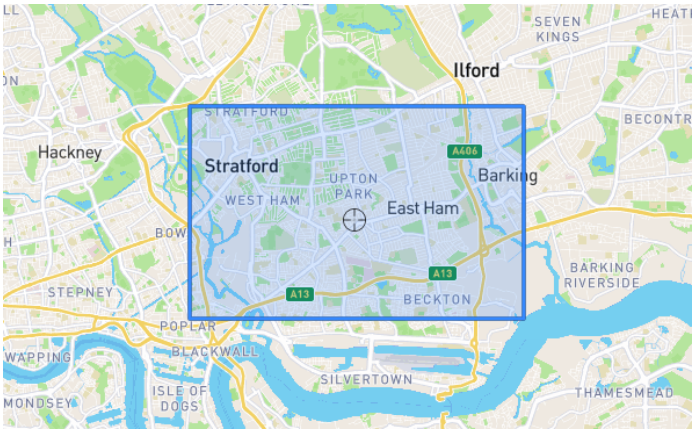

Figure 6: Bounding box(bbox). Screenshot taken from bboxfinder.com (bboxfinder.com contributors 2019)

The Overpass API allows to filter data by location and one of the ways to do it is via a bounding box(bbox): a type of filter that obtains data within an area. For this project, all OSM data was acquired with a bbox of the value "`51.51260436919461,-0.013602060765980596,51.55173130930535,0.08524790329606169`" [4], that contains regions of Stratford(London) and East Ham(London). The reason for this locations is due to the author's knowledge of the location, which will be used to assess if a generated route is safe or not.

Geographic street data and points of interest were fetched by doing HTTP requests to the Overpass API, with the body following the Overpass Query Language(QL) format. After downloading the data from Overpass, they are converted to GEOJSON format(Butler et al. 2016) using the `osm2geojson` package(osm2geojson contributors 2022) and then splitting the file into multiple ones by geometry type(e.g: points, linestrings, polygons).

- Street data: Its QL query(See Appendix A.1) relates to every path that is, or is related to, a path traversable by foot. Additionally, paths with the attribute "name" as "north_st_john_church" and "the_grove_to_broadway" were added to fill missing paths between streets.

- Points of Interest: The motivation to fetch points of interest(See Appendix A.3 for the query) are based on the recommendation by the police forces to walk on paths/areas with high busyness level(Lancashire Constabulary 2022)(Nottinghamshire Police 2022). Thus, based on google implementation(Google 2022a), the proposed cost algorithm uses stores, restaurants, cafes, and more to determine its busyness level.

.

The `ogr2ogr` tool (GDAL/OGR contributors 2022b) (GDAL/OGR contributors 2022a) was used to import the OSM geometries and features/attributes to the PostgreSQL database. The final tables' Schema(The unusued

[4]BBox can be visualised here: `http://bboxfinder.com/#51.512604,-0.013602,51.551731,0.085248`

columns are ommitted) can be seen in Figures 7 and 8. The column `id` refers to the OSM ID, the `gid` is just a unique id that auto increments.

edges		
PK	gid	int
	id	int
	highway	varchar
	lit	varchar
	sidewalk	varchar
	sidewalk:right	varchar
	sidewalk:left	varchar
	access	varchar
	source	int
	target	int
	geom	geometry (llinestring)
	...	...

Figure 7: Crime street data was imported to the database as a table called **edges**

busy_sources_points		
PK	gid	int
	id	int
	geom	geometry (point)
	...	...

Figure 8: Points of interest were imported to the database as a table called **busy_sources_points**.

3.5 Street Network Setup

To calculate the safest route, a set of nodes/vertices and edges connecting them is required to represent a graph structure(Carlson, S. C. 2022), which is essential to run the Dijkstra Algorithm. The edges will be the OSM street data. To make sure every intersection has a separate road segment, the `pgr_nodeNetwork` function (Ribot 2021) was used(See query below), creating a new edges table called `edges_noded`.

```
SELECT pgr_nodeNetwork('edges', 0.00001, 'gid',
    'geom');
```

All the attributes presented in Fig.7 were copied to the `edges_noded` table and `pgr_nodeNetwork` created 2 new attributes/columns:

- old_id: The original edge(`edges` table) ID.
- sub_id: The segment identifier related to the original edge. Due to `pgr_nodeNetwork`, an edge can have multiple segments.

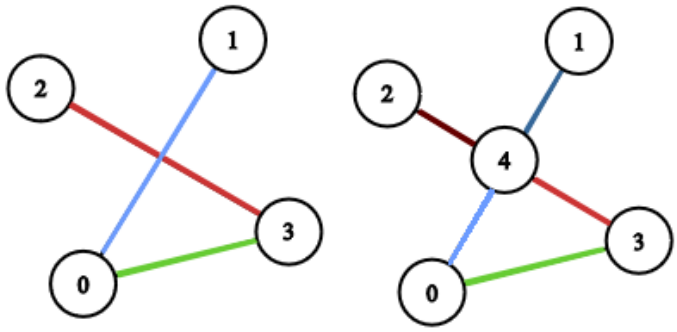

Figure 9: Graph/Street network before(left side) and after(right side) being "noded"(Ribot 2021) by **pgr_nodeNetwork**. Comparing the noded version and the original, there is an additional node(number 4) and two more edges/segments: From node 2 to 4 and from 4 to 1. This makes possible traversing from point 0 to 2, which in the original was not. It's important to note that the nodes are virtual. The function just creates edges.

The column `geom_length` was created to populate it with the segment/street distance in meters, using the query below:

```
ALTER TABLE edges_noded ADD COLUMN IF NOT
    EXISTS geom_length FLOAT8;
UPDATE edges_noded SET geom_length = ST_Length(
    ST_Transform(geom, 4326)::geography);
```

edges_noded		
PK	id	int
	old_id	int
	sub_id	int
	highway	varchar
	lit	varchar
	sidewalk	varchar
	sidewalk_right	varchar
	sidewalk_left	varchar
	access	varchar
	source	int
	target	int
	geom	geometry (llinestring)
	geom_length	double precison
	...	...

Figure 10: Table "edges_noded" Schema

To create the graph's vertices and effectively create the final topology, the pgRouting function `pgr_createTopology` (pgRouting Contributors 2021a) is utilized:

```
SELECT pgr_createTopology('edges_noded',
    0.00001, 'geom', 'id');
```

After execution, it creates a table containing the vertices/nodes of the street network called `edges_noded_vertices_pgr`.

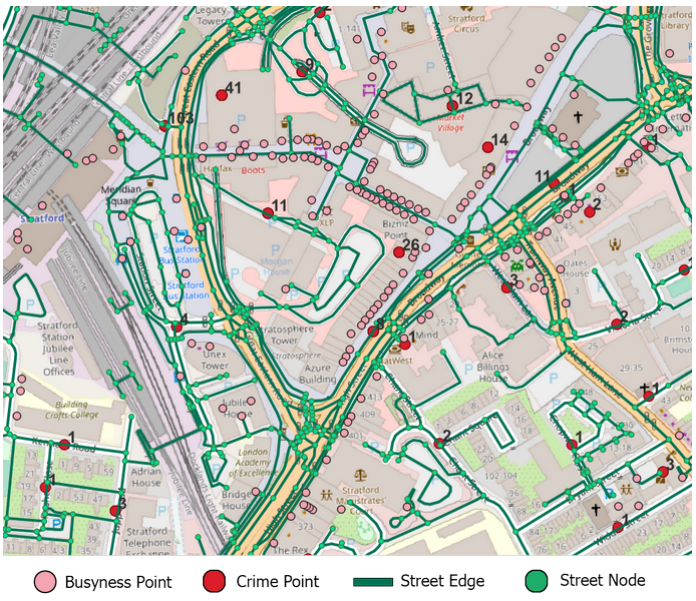

Figure 11: Street Network Topology. The "Street Edge" and "Street Node" refer to the tables "edges_noded" and "edges_noded_vertices_pgr" respectively.

3.6 Proposed Cost Algorithm

The proposed algorithm has parameters defined based on police recommendations of what is considered a safe walking route, and also crime density. The cost result is interpreted as a risk score (Rs). This algorithm will determine the street/edge cost for the Dijkstra Algorithm, which its value is proportional to the risk level of the street.

The parameters' final weights were defined after many attempts of generating a route that would be considered safe by the author's knowledge of which roads are safe. He lives in Newham and goes to the Stratford station frequently, so while the method has a heavy bias, it allows to confirm that some generated routes are indeed safe.

In the remaining sections, a function called `ST_Dwithin` (POSTGIS 2022d) will be mentioned. It works differently if its arguments are of type `geometry` or `geography`. For `geometry` types, the `radius` unit is in **meters**. For `geography` types, the unit depends on the SRID of the arguments. Both $edge_{geom}$ and b_{geom} use the 4326 SRID, so the radius unit will be in degrees. In order to convert meters to degrees and not be computational intensive, an approximation of values was done based on the information provided by U.S. Naval Academy (2019), setting a custom unit($mdeg$) that refers to meters expressed as approximated degrees(deg):

$$\frac{mdeg}{deg} = 1000000 \iff 1mdeg = 0.00001deg \quad (1)$$

Variables present in further sections:

E = Set of Edges(Table `edges_noded`)

V = Set of Vertices
(Table `edges_noded_vertices_pgr`)

$CrimesStreet$ = Set of Crimes
(Table `crimes_street`)

$CurrCrimes$ = Set of Current Crimes
(Table `current_crimes`)

$BusySources$ = Set of Busy Sources
(Table `busy_sources_points`)

$edge$ = Element of E being traversed by the Dijkstra Algorithm

3.6.1 Cost calculation

The cost algorithm is represented by the following(See Appendix A.7 to see the SQL query):

$$\begin{aligned} Rs = cost(edge, weights...) = \\ & lengthCost(edge) \\ & + crimeCost(wCrime, edge) \\ & + notLitCost(wNotLit, edge) \\ & + serviceTypeCost(wServiceType, edge) \\ & + lackOfBusySources(wLackBusy, edge) \\ & + roadsWStoredSidewalksCost(edge) \\ & + currentCrimesCost(wCurrCrimes, edge) \\ & + privateCost(edge) \\ & + susRoadCloseToServiceCrimeCost(\\ & \quad wSusRoadCloseToService, edge) \\ & + nearParkAisleCost(wServiceType, edge) \end{aligned} \quad (2)$$

The algorithm can be split into 10 parts:

- $lengthCost$ Distance/Size of the road.
- $crimeCost$ Number of crime points there are near the edge/road. Weight variable: $wCrime$
- $notLitCost$ Increase Cost if the street is not lit.(Weight variable: $wNotLit$).
- $serviceTypeCost$ Cost of street being one of values (alley, parking_aisle). (Weight Variable: $wServiceType$)
- $lackOfBusySourcesCost$ Cost of not having sources of busyness(e.g shops,offices) near the edge/street. (Weight Variable: $wLackBusy$)
- $roadsWStoredSidewalksCost$ If the current street being analysed has a sidewalk that is in the dataset, then its cost is increased to prioritise the sidewalks.
- $currentCrimesCost$ Cost for crimes that are happening near the street. (Weight Variable: $wCurrCrimes$)

- *privateCost* Private streets cost.

- *susRoadCloseToServiceCrimeCost* Additional crime cost for streets that have *highway* $\in$ "*pedestrian*" and if they are close to a street of `highway="service"`. (Weight variable: *wSusRoadCloseToService*)

- *nearParkAisleCost* - Roads near parking aisles are penalised. (Weight Variable: *serviceTypeCost* (Same as *serviceTypeCost*))

3.6.2 Crime Cost

The Crime Points Cost is relative to crime that happened near the road being analysed. For each edge that is analysing, it fetches crimes points belonging to (*CrimesStreet*) where the road is present within a $100m$ radius(See Fig.12 for example).

$$CrimeR = 100m \tag{3}$$

A $200m$ radius is recommended for crime prediction (Okabe, Satoh, and Sugihara 2009) (Khalid, Shoaib, Qian, Rui, A. Bari, et al. 2018), but depending on the context, a range between 100-300m is valid(Khalid, Shoaib, Qian, Rui, A. Bari, et al. 2018)(Porta et al. 2009) for calculating population density. The $100m$ value gives a coarse result(Khalid, Shoaib, Qian, Rui, A. Bari, et al. 2018). However, to the advantage of this project, it allows to distinguish risk between residential streets that are close to each other, therefore prioritising safer ones.

The motive of using of using max quantities was to increase performance while keeping the result valid. However its values might need to change due the amount of data. Our final dataset is a month worth of crime, if using a dataset that utilises 2 months, then these limits would need to doubled. There was an average of 2.55 crimes(See Appendix A.6 for the query) per Snap Point. So a we chose a limit of 20 crimes around a edge to be the threshold.

$$CrimePointsNearEdge(edge) = \{cst \in CrimesStreet \mid ST\_DWithin(edge_{geom}, cst_{longlat\_point}, CrimeR)\} \tag{4}$$

$$crimeCost(edge) = edge_{geom\_length} \cdot wCrime \cdot max(|CrimePointsNearEdge(edge)|, 20) \tag{5}$$

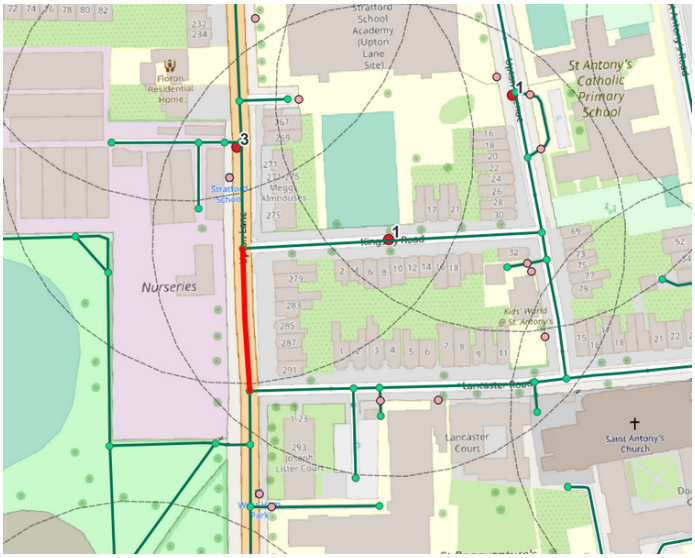

Figure 12: The highlighted edge/road in red has 4 crimes nearby(See Appendix A.5 for the SQL query). The circles with dashed lines represent the $100m$ radius.

3.6.3 Not Lit Cost

The cost for a road not being lit was implemented due to the police recommendation of avoiding unlit streets(Lancashire Constabulary 2022)(Nottinghamshire Police 2022). It was decided to opt-in `NULL` values for the `lit` attribute because some roads might be lit, but the confirmation is not in OSM.

$$isLit(litValue) = [litValue \in \{\text{'yes'}, NULL\}] \tag{6}$$

$$notLitCost(wNotLit, edge) = edge_{geom\_length} \cdot wNotLit \cdot !isLit(edge_{lit}) \tag{7}$$

3.6.4 Service Type Cost

Lancashire Constabulary (2022) recommends avoiding parking isles and narrow roads such as alleys, so we penalise roads with `service` type of `parking_aisle`, `alley`.

$$serviceTypeCost(wServiceType, edge) = edge_{geom\_length} \cdot wServiceType \cdot [edge_{service} \in \{\text{"service"}, \text{"alley"}\}] \tag{8}$$

3.6.5 Lack of Busyness Sources Cost

To prioritise busy areas/roads as suggested by the police, the proposed algorithm searches for restaurants, cafes, and more that are near of the edge. For each edge that is analysing it fetches "busy" points belonging to (*BusySources*) where the road being analysed is present within a 15 m radius.

$$busyPointsNearEdge(edge) = \{b \in BusySources \mid [ST\_DWithin(edge_{geom}, b_{geom}, 15\ mdeg)]\} \tag{9}$$

$$lackOfBusySourcesCost(wLackBusy, edge) = edge_{geom\_length} \cdot (wLackBusy - (\frac{wLackBusy}{15} \cdot max(|busyPointsNearEdge|, 15))) \quad (10)$$

3.6.6 Roads with stored Sidewalks Cost

$$HighwaysWithStoredSidewalksCost = edge_{geom\_length} \cdot 0.03 \quad (11)$$

3.6.7 Current Crimes Cost

Beyond the crime data from data.police.uk (2022b), a another type of crimes were added as "Current Crimes". Current Crimes is a implementation of this project that serves to save crimes and that are happening at the moment, which are reported by the users of our web application. It's very useful to have this kind of data, because by having crimes that are happening, it allows the algorithm to generate a high cost for roads nearby, therefore urgently avoiding the street with elevated risk. It follows the same logic as the `Crime Cost`, but it uses $150m$ instead of $100m$: For each edge that is analysing, it fetches crimes points belonging to ($CurrCrimes$) where the road/edge in question is present within a $150m$ radius.

$$CurrentCrimesNearEdge(edge) = \{currCrime \in CurrCrimes \mid ST\_DWithin(edge_{geom}, currCrime_{geom}, 150m)\} \quad (12)$$

$$currentCrimesCost(wCurrCrimes, edge) = edge_{geom\_length} \cdot wCurrCrimes \cdot max(|CurrentCrimesNearEdge(edge)|, 5) \quad (13)$$

3.6.8 Private Cost

To avoid trespassing/evading private roads, a cost for roads with `access=private` was added.

$$privateCost(edge) = edge_{geom\_length} \cdot 2 \cdot [edge_{access} \in \{\text{"private"}\}] \quad (14)$$

3.6.9 Suspicious Roads Close to Service Roads

Due to a road close to Stratford Centre that connects to a road of highway type of `service`, being narrow and feeling dangerous to the author when he walked there, this cost was added. To be considered close to a service road, the start of the edge(`ST_StartPoint` (POSTGIS 2022f)) or the end of it(`ST_EndPoint` (POSTGIS 2022e)) need to be within a $35m$ range of a service road. For its crime calculation, it fetches crimes points belonging to ($CrimesStreet$) where the road/edge in question is present within a $25m$ radius from the crime point. The reason behind the radius being less than the Crimes Cost equation is that this cost is aimed for narrow streets(e.g pedestrian, service) so a smaller values makes sure that only searches the mentioned types of street.

$$CrimePointsSusRoads(edge) = \{cst \in CrimesStreet \mid ST\_DWithin(edge_{geom}, cst_{longlat\_point}, 25m)\} \quad (15)$$

$$
\begin{aligned}
&isSuspitiousRoad(edge) = \\
&\quad [edge_{highway} = pedestrian\ AND \\
&\quad \{otherEdge \in E\ \mid otherEdge_{highway} = service\ AND \\
&\quad (ST\_DWithin(otherEdge_{geom}, \\
&\qquad geometry(ST\_StartPoint(edge_{geom})), 35m) \\
&\quad OR \\
&\quad ST\_DWithin(otherGeom_{geom}, \\
&\qquad geometry(ST\_EndPoint(edge_{geom})), 35m))\}]
\end{aligned}
\tag{16}
$$

$$
\begin{aligned}
&susRoadsCloseToServiceCost(wSusRoadCloseToService, \\
&\quad edge) = \\
&\quad [isSuspitiousRoad] \cdot (edge_{geom\_length} \cdot \\
&\quad wSusRoadCloseToService \cdot \\
&\quad max(|CrimePointsSusRoads(edge)|, 20m))
\end{aligned}
\tag{17}
$$

3.6.10 Roads near Vehicle Park Aisles

As a complement to the Section.3.6.4, every road that is close to a parking aisle is penalised.

$$
\begin{aligned}
&isCloseToParkingAisle = \\
&\quad [edge_{service} \neq \text{parking\_aisle}\ AND \\
&\quad \{otherEdge \in edges\ \mid \\
&\quad otherEdge_{service} = \text{parking\_aisle}\ AND \\
&\quad ST\_DWithin(otherEdge_{geom}, \\
&\qquad ST\_Centroid(edge_{geom}), 20mdeg)\}]
\end{aligned}
\tag{18}
$$

$$
\begin{aligned}
&roadsNearParkingAislesCost(wServiceType, \\
&\quad edge) = \\
&\quad [isCloseToParkingAisle] \cdot \\
&\quad edge_{geom\_length} \cdot wServiceType
\end{aligned}
\tag{19}
$$

3.7 Assessing Time Complexity

Unfortunately, in order to confirm the time complexity of the algorithm, a deeper research on the implementation of R-Tree indexes by PostGIS would be required. So in this section, we'll only declare the research that was done.

We used the geometry(The PostgreSQL Global Development Group 2022b) and geography(POSTGIS 2022b) data types with the GiST index (The PostgreSQL Global Development Group 2022a)(Hellerstein 1999) to store the geographic data. The PostgreSQL's implementation (The PostgreSQL Global Development Group 2022a) of the GiST is made by Sigaev and Bartunov (2002). They provide many variations of the Gist index such as R-Tree and B-Tree implementations. PostGIS uses R-tree for indexing geometries (POSTGIS 2022c).

It's not guaranteed, but according to this research, PostGIS uses the R-Tree implementation of the GIST Index. So if an research about this implementation is done, then the time complexity could have been assessed.

3.8 Generating a safe route

With the cost algorithm set, we use pgRouting's `pgr_dijkstra(edges, start_vid, end_vid, directed)` function (pgRouting Contributors 2021b) to run the Dijkstra's algorithm using the data from the table `'edges_noded'` as edges and `edges_noded_vertices_pgr` as vertices. The result will be the Safe Route. Please see Appendix A.8 for the query.

3.9 Web Application

As a proof of concept, a web application was created to help people plan their safe route.

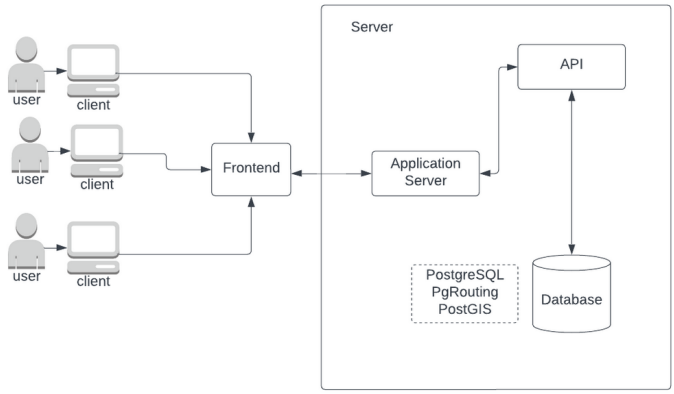

Figure 13: Web Application diagram. It follows a client-server architecture.

- Each user utilizes one or more devices that act as **clients**
- Each client interacts with a **frontend** provided by the Application Server and each client has their own channel of communication with the Application Server.
- The Application Server handles the requests of the client. When the client needs data that is not present on the frontend, the Application Server dispatches client's request to the Application Program Interface(API).
- The **API** validates and filters the user's request if necessary. After that, it will fetch data from the database, format it, and send it to the client.
- The **database** is a PostgreSQL instance with PostGIS and pgRouting as extensions which are responsible for the route generation.

As a warning to the reader/user, there are bugs in the web application, especially when generating a route. If it happens, repeat the steps on Section 3.9.3.

3.9.1 Backend/Server

The backend was created using Node.js v14.17.6 (OpenJS Foundation 2022) with the express.js framework(OpenJS Foundation 2017).

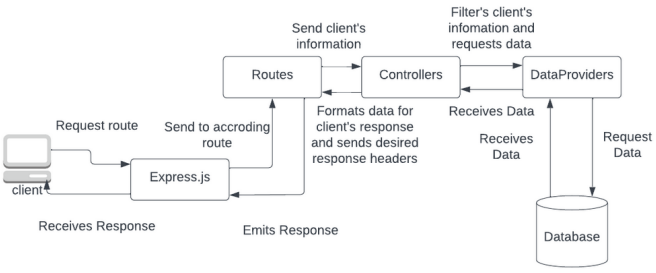

Figure 14: Application Server Architecture

- Express.js is the Node.js framework. It makes creating a HTTP server easier to develop.
- Routes: Every route establishes an URL endpoint for a client to interact(e.g `http://127.0.0.1/api/routeplan`, `http://127.0.0.1/api/current_crimes`).
- Controllers: Every controller is responsible for the business logic.
- Data Providers: Methods that retrieve data.
- Database: Where data is stored and, route is generated.

3.9.2 Frontend

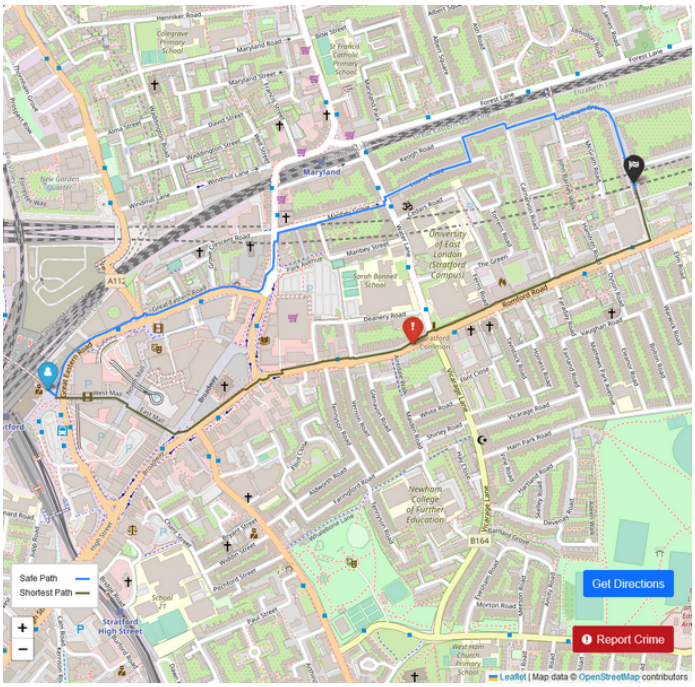

Figure 15: Web Application - Safest Route App

The frontend is responsible for the user's interaction. The technology used are the standard Javascript, HTML and CSS. In terms of major libraries, Leaflet(Vladimir Agafonkin 2022) was used for map rendering and interactivity.

For User Experience, the frontend is responsive, therefore it's usable in large and small screens. Additionally, the way the user picks the start and destination position markers change depending if it is a touch device or not. If it is not a touch device(e.g desktop computer), the user can drag directly the marker or click in the map to choose a new position. When it is a touch device the marker is fixed on the middle of the screen and the user can drag the whole map to change marker's position. This makes easier for the user to navigate because the map has more area of interactivity than the marker. If it were the same method as the non-touch devices, the user would need to be very precise in order to drag the marker.

3.9.3 Generating and displaying a safe route

The user can set a starting and destination location by either dragging individually or clicking on the "Get Directions" button.

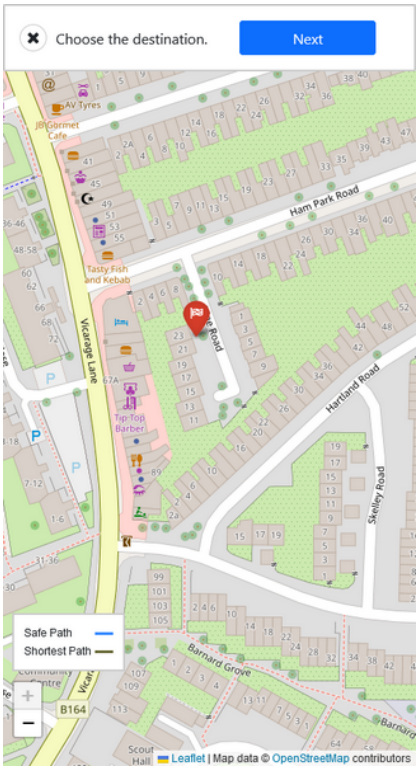

Figure 16: Picking Destination location after clicking the "Get Directions" button. When the the user's clicks "Next", it prompts for the starting position (See Fig.17

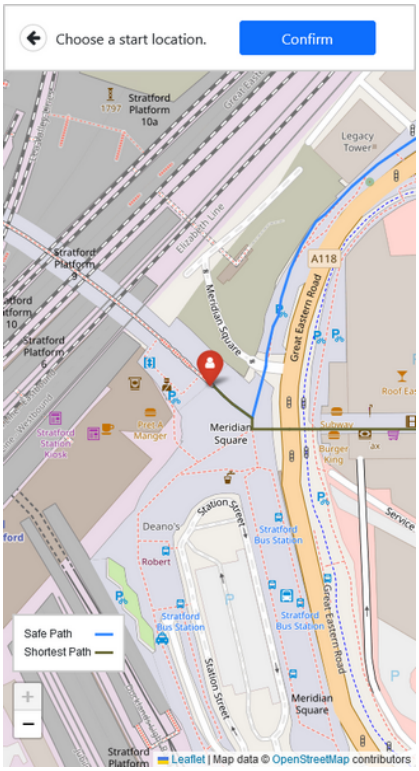

Figure 17: Picking starting location.

After clicking the "Confirm" button(Fig.17), the `client` will request a safe route to the `API`. The API returns the route in GEOJSON format, which can be read by Leaflet and render on the map.

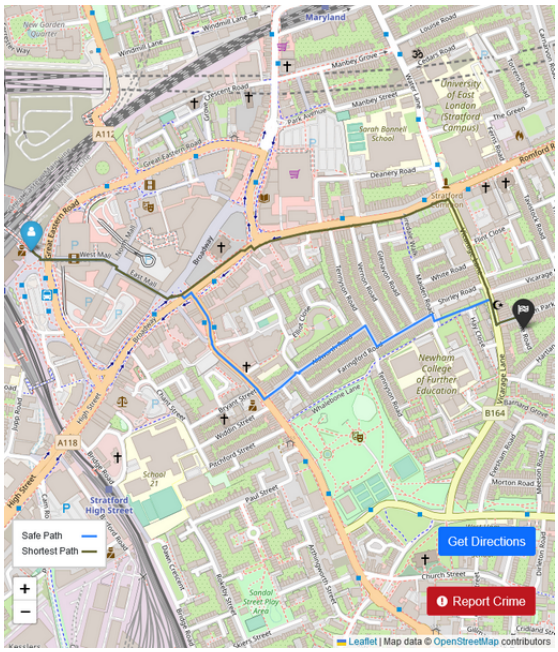

Figure 18: Generated Route

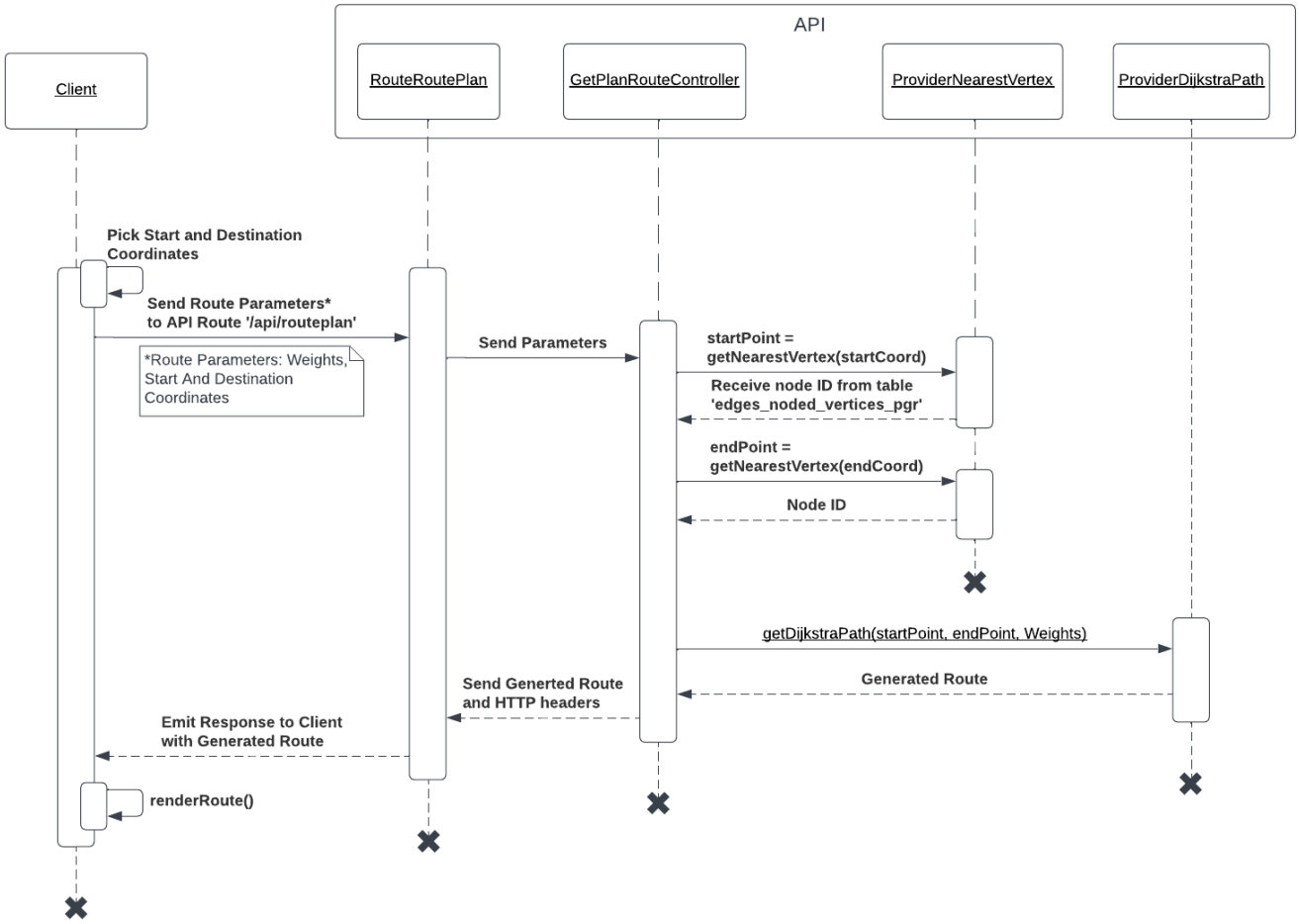

Figure 19: Sequence Diagram for generating a route and displaying it.

3.9.4 Reporting a Crime

The user can report a crime by clicking on the "Report Crime" button (Fig.15). The user then can select the crime location with same picker experience as with generating a route(Fig.20).

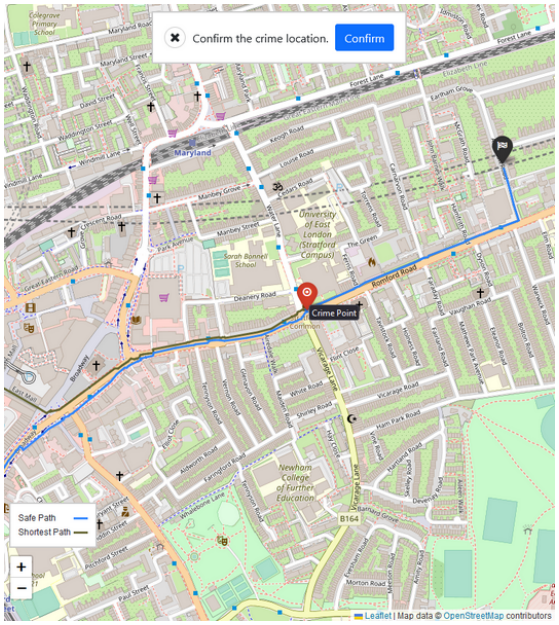

Figure 20: Report crime selection

3.9.5 Deleting a Current Crime

The user can delete a current crime by clicking on it, and then clicking on the "trash bin" icon.

This feature would not be accessible to every user, it would probably be delegated to users with high privileges such as police. A police member could delete a crime after dealing with it. Since this is just a proof-of-concept, the feature is available to anyone for testing purposes.

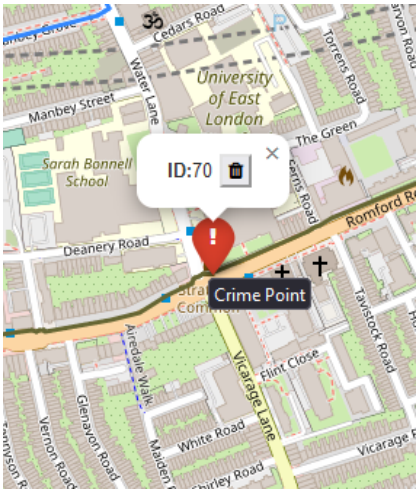

Figure 21: Deleting Current Crime. After clicking on the current crime marker, a popup will be shown. In there, a button with a trash bin icon is present, which once clicked, it will delete the current crime.

4 Experiments

The main goal of this project is to generate safe pedestrian route, so in this section it will be explained how the safety of these paths were assessed.

4.1 Experimental settings

For this project the algorithm performance is bound by the CPU and RAM. Two computers were used for testing:

Computer 1

CPU: Intel i5-4460

GPU: AMD R9 380

RAM: 16GB Operating System: Windows 10

Computer 2

CPU: Intel i5-4200U

GPU: NVIDIA Geforce 740M

RAM: 4GB Operating System: Windows 10

It needs to be prefaced that the algorithm is slow. For **Computer 1** it takes around 10 seconds to generate a safe route. **Computer 2** takes around 30 seconds.

To assess the safety of the routes provided by the algorithm, many trips($Trip$) were created to test multiple case scenarios. Where each one being a set of a start($SPos$) and destination($DPos$) point locations.

$Trip_i = (SPoint_i, DPoint_i)$

$SPoint, DPoint = (latitutde, longitude, NID)$

The table below refers to the trips tested. The `NID` refers to vertex/node ID in the table `edges_noded_vertices_pgr`. The `Lat,Long` refers to Latitude and Longitude of the vertex ID. The Description gives a brief explanation of where the vertices are located.

Table 1: Trips to test

i	SPoint(NID)	SPoint(Lat, Long)	DPoint(NID)	DPoint(Lat, Long)	Description
1	9836	51.5412285, −0.0027388	2575	51.5455569, 0.0153327	Stratford Station to Atherton Road
2	9836	51.5412285, −0.0027388	721	51.5371811, 0.0089465	Stratford Station to Densham Road
3	1359	51.5412338, 0.0139709	1389	51.5369718, 0.0231854	Ham Park Road to Plashed Road
4	16317	51.5338416, 0.0628981	824	51.5238595, 0.0575292	Barking Road to Highstreet South
5	11372	51.5428719, −0.0005575	721	51.5455569, 0.0153327	Great Eastern Road to Atherton Road

The proof-of-concept Web Application and the GIS software QGIS will serve to visualise the paths. The latter will be helpful to analyse more deeply each route.

For each trip, at least two routes will be presented:

- Safe Route: containing the final weights that result on paths that seem safe.
- Shortest Route: every weight is set to 0, so it will only count the road distance, giving on the fastest route to do the trip. This route is ideal for comparison because it simulates the default behaviour of popular navigation applications such as Google Maps, where its main goal is to determine the fastest route, which might not be the safest.

Table 2: Safe Route Weights

Weight	Value
$wCrime$	0.07
$wNotLit$	5
$wServiceType$	5
$wLackBusy$	5
$wCurrCrimes$	20
$wSusRoadsCloseToService$	20

Table 3: Shortest Route Weights

Weight	Value
$wCrime$	0
$wNotLit$	0
$wServiceType$	0
$wLackBusy$	0
$wCurrCrimes$	0
$wSusRoadsCloseToService$	0

For every time a change of weights is mentioned, it is meant to replace the original weight present in the "Safe Route" configuration (Table 2).

4.2 Evaluation criteria

If the generated paths change according to the change of parameters and their weights, it means the the route is being well generated. Additionally the author will use his knowledge of some roads to confirm their level of safety and through visual analysis see if the routes follow the police guidelines.

4.3 Results

4.3.1 Trip 1 - Stratford Station to Atherton Road

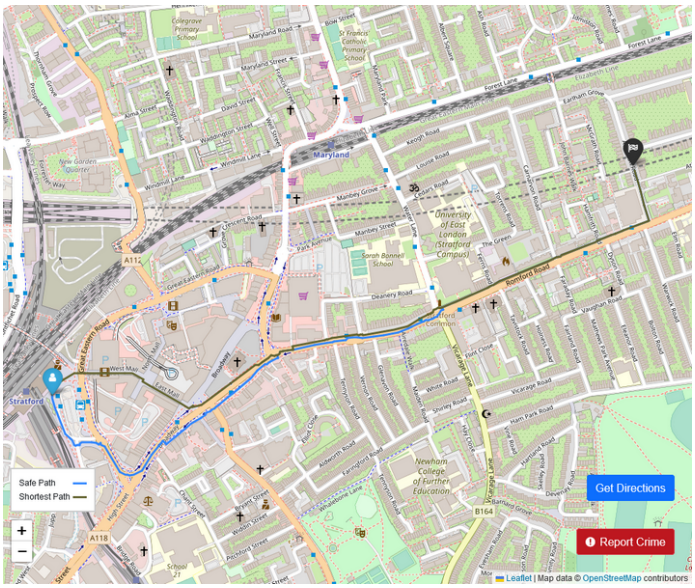

Figure 22: Trip 1 is a route from Stratford Station to Atherton Road.
Safe Route, Shortest Route

By analysing Fig.23, the major difference between the safe route and the shortest route is that the prior one does not go into the Stratford Centre. Even though it's a busy area, the amount of crime makes not viable to go there(See Fig.24).

If the cost or weight($wLackBusyPoints$) of not having busyness points around is not considered($wLackBusyPoints = 0$), the algorithm prioritises a residential area creating a great detour (Fig.23, Route), that is not considered as safe and also not practical.

In Fig.25, it describes the hypothetical situation of a user reporting a crime that is happen-

ing "now" at the intersection between Romfoard Road and Water Lane(Coordinates: 51.5429054772399, 0.009377002716064455). It can be observed that comparing to Fig.22, the algorithm avoids Romfoard Road and chooses a residential area to effectively avoid the crime, therefore preventing possible harm to the pedestrian.

This route satisfies the criteria of prioritising busy areas as the police recommends and takes the crime data in consideration.

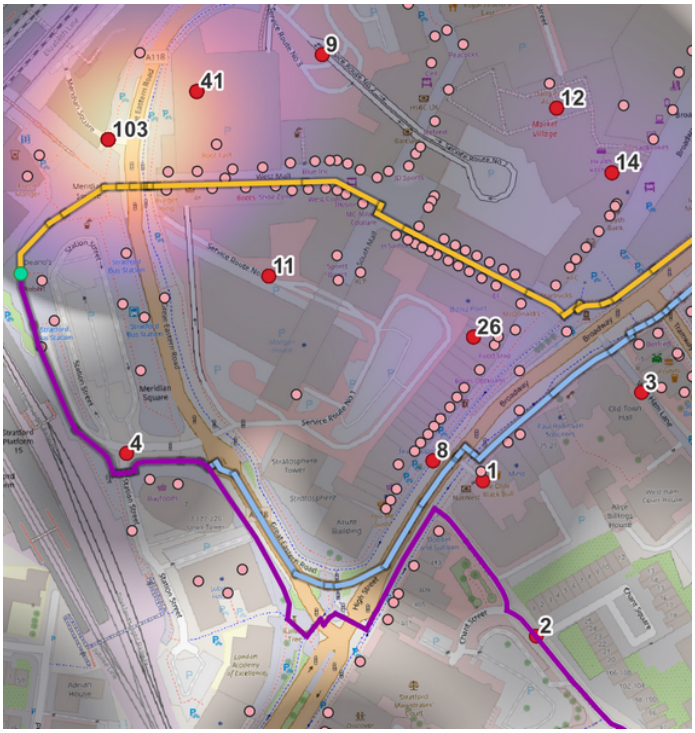

Figure 24: The safe route goes around Stratford Centre due to the crime rate. As it can be seen in the crime heatmap, the safe path prioritizes the less concentrated area of crime.
Safe Route, Shortest Route, Busyness Points, Safe Route with $wLackBusyPoints = 0$,

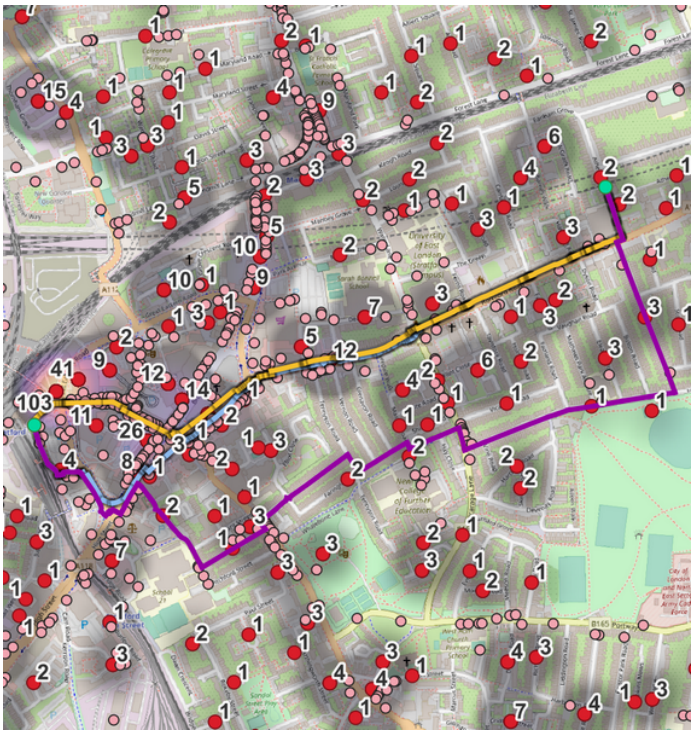

Figure 23: Trip 1 in QGIS.
Safe Route, Shortest Route, Busyness Points, Safe Route with $wLackBusyPoints = 0$,

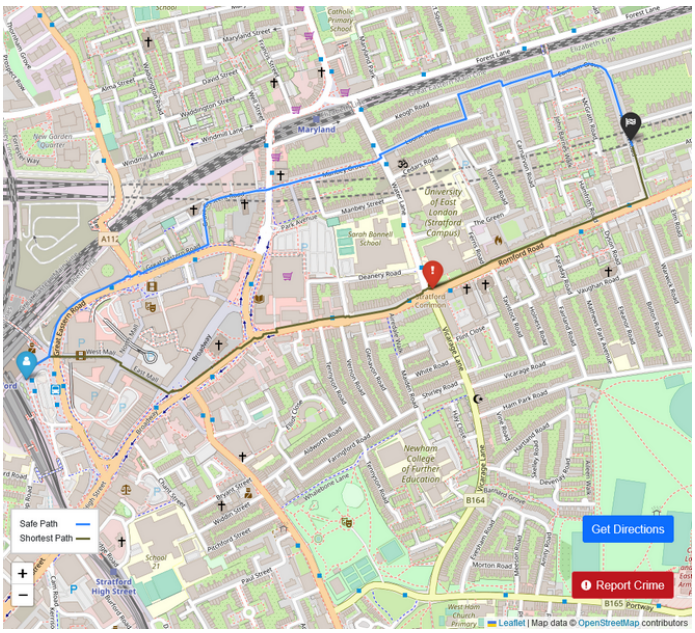

Figure 25: Safe Route algorithm adjusting to a crime that is happening at the moment.
Safe Route, Shortest Route, Crime happening at the moment.

4.3.2 Trip 2 - Stratford Station to Densham Road

Figure 26: Trip 2 is a route from Stratford Station to Dansham Road. Scheenshot from the Web Application
Safe Route, Shortest Route

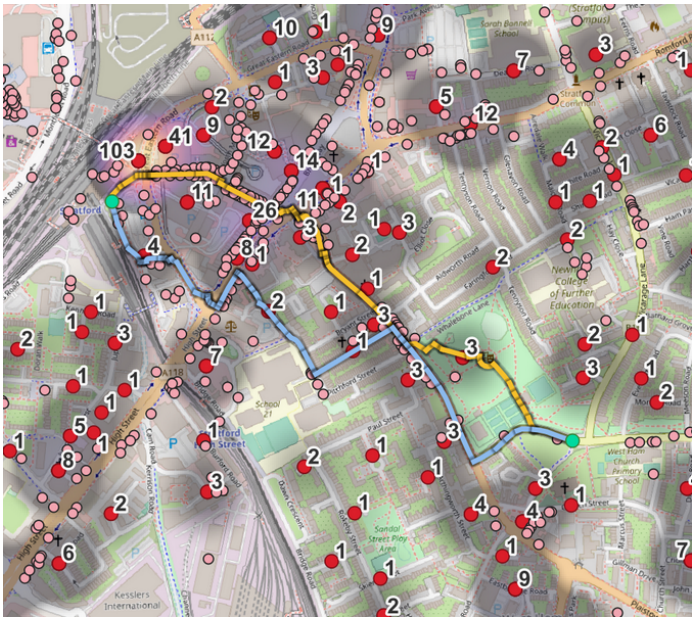

Figure 27: Trip 2 in QGIS.
Safe Route, Shortest Route, Busyness Points

In this trip we want to emphasize the cost of a road not being lit. It prioritizes going around Stratford Centre like Trip 1 and chooses to go around the Stratford Park(See Fig.28) instead of going inside due not being lit. This trip demonstrates it's considering unlit roads to be avoided, as the police suggests.

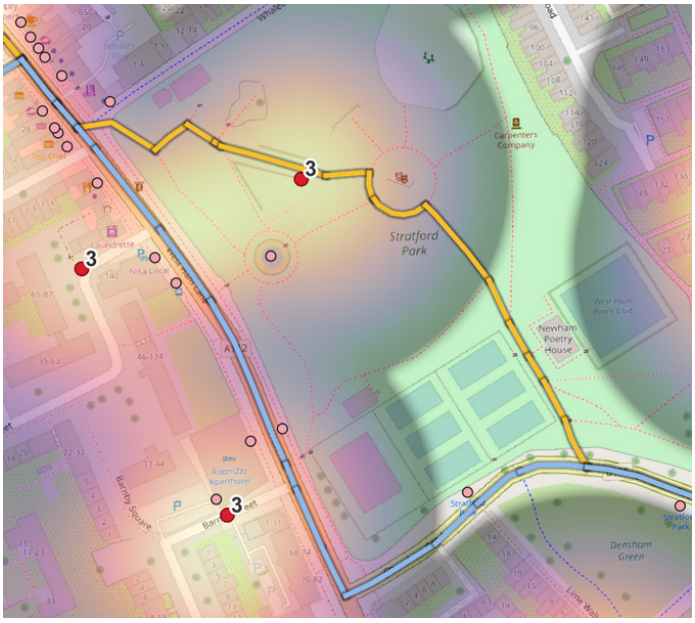

Figure 28: Stratford Park detour.
Safe Route, Shortest Route

Figure 29: Unlit roads in the Stratford Park: The reason for the safe route not choosing this path as viable.
Safe Route, Shortest Route Unlit roads

4.3.3 Trip 3 - Ham Park Road to Plashed Road

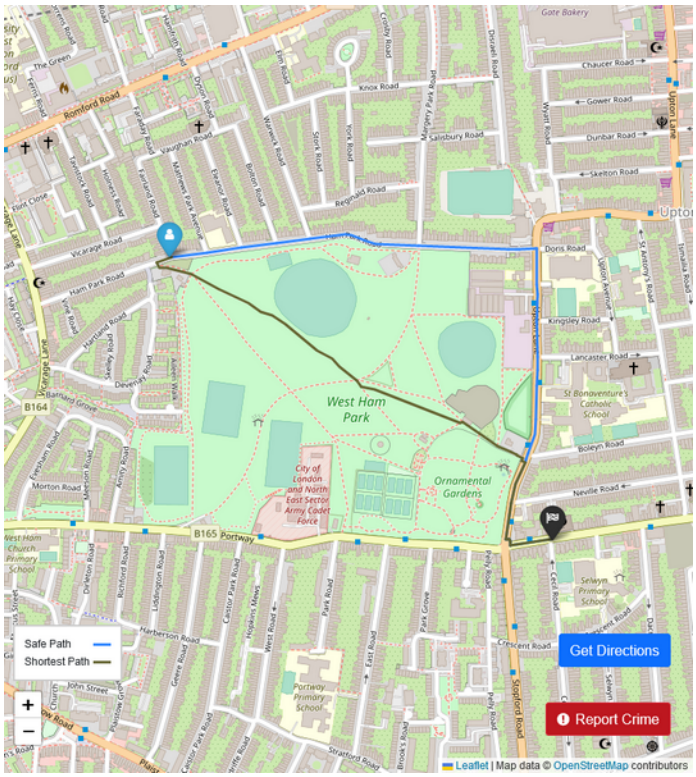

Figure 30: Trip 3 is a route from Ham Park Road toPlashed Road. Scheenshot from the Web Application
Safe Route, Shortest Route

Extending the focus on unlit roads, we've tested a trip that included a larger park, namely West Ham Park. It does the intended detour as `Trip 2` (See Fig 31).

This test was also a important element to the development of the final weigh $wNotLit = 5$. In earlier stages of development, the cost for unlit roads was $wNotLit = 2$, which was sufficient for trips such as `Trip 2` where the roads are small. However, when using $wNotLit = 2$ in this trip, it made the algorithm still consider the park as viable option (See The Purple Route in Fig 31). When this was noticed, the weight was changed to $wNotLit = 5$.

Figure 31: Trip 3 in QGIS.
Safe Route, Shortest Route, Busyness Points, Safe Route with $wNotLit = 2$

Figure 32: Unlit roads in the West Ham Park: The reason for the safe route not choosing this path as viable.
Safe Route, Busyness Points, Safe Route with $wNotLit = 2$, Unlit Roads

4.3.4 Trip 4 - Barking Road to High Street South

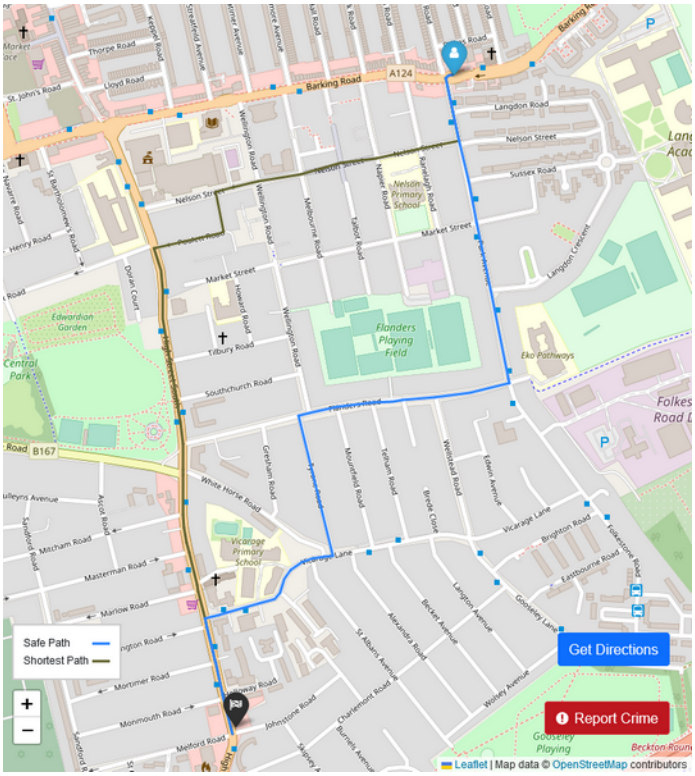

Figure 33: Trip 4 is a route from Barking Street to Highs Street South. Screenshot from the Web Application
Safe Route, Shortest Route

Figure 34: Trip 4 in QGIS. Due to the crime density, the safe route prioritises the residential area.
Safe Route, Shortest Route, Busyness Points

This trip demonstrates that the algorithm under certain circumstances prioritises residential areas. Since there is a clear gap in crime rate between Tyrrone Road and Park Avenue, the algorithm choose those locations as the safest paths. This can be supported by the fact that when removing the consideration of crime density($wCrime = 0$) out of the Cost Algorithm, it generates the same route as the "Shortest".

Even though the Safe Route avoids crime as expected, it can also be a possible issue of the crime prediction calculation(Section 3.6.2). It uses a small radius of $100m$, which might might give granular results (Khalid, Shoaib, Qian, Rui, A. Bari, et al. 2018), thus leaving segments of roads unchecked.

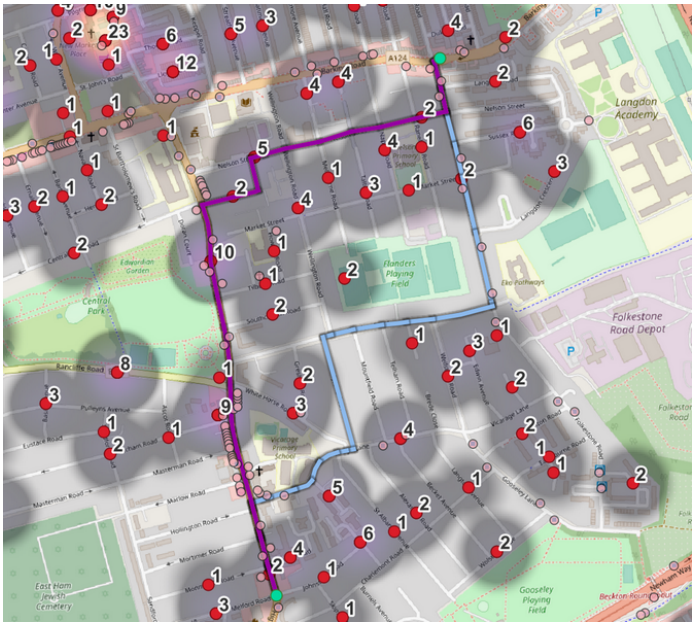

Figure 35: Trip 4 Safe Route without considering crime rate($wCrime = 0$). Without it, it is the same as the Shortest Route.
Safe Route, Safe Route with $wCrime = 0$, Busyness Points

4.3.5 Trip 5 - Great Eastern Road to Atherton Road

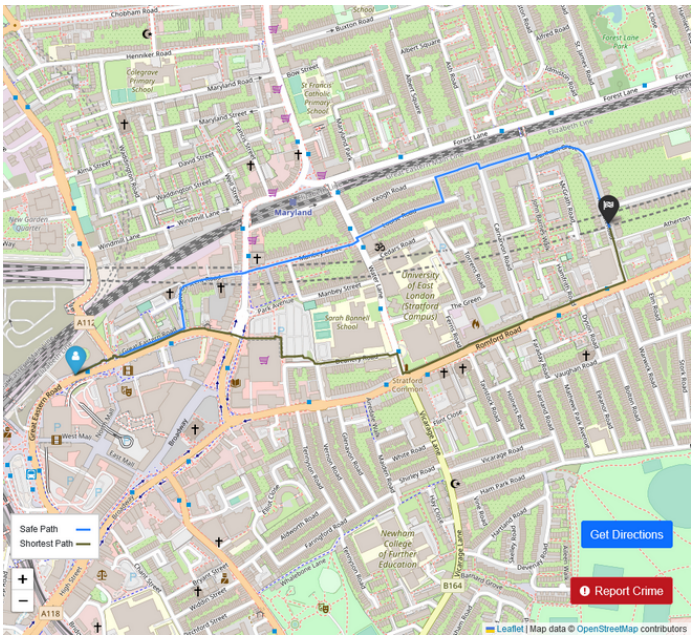

Figure 36: Trip 5 is a route from Great Eastern Road to Atherton Road. Screenshot from the Web Application
Safe Route, Shortest Route

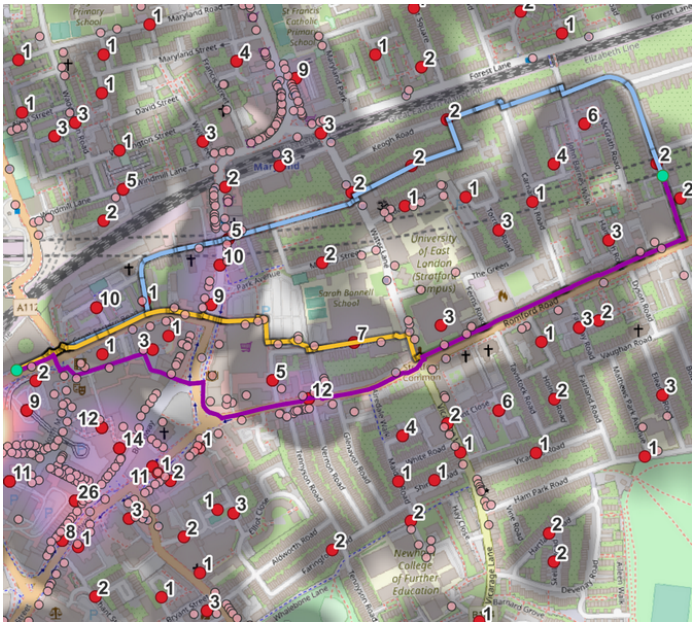

Figure 37: Trip 5 in QGIS.
Safe Route, Busyness Points, Safe Route with $wSusRoadCloseToService = 0$

This is the trip that motivated the creation of adding cost of crime for pedestrian(`OSM's highway=pedestrian`) roads close to service ones (`OSM's highway=service`) roads (Section 3.6.9). Without that cost, the safe route would go through a narrow road(Purple Route in 37), namely, Salway Place(Coordinates: 51.54277219899664, 0.002983459231920582)(Fig.39). The police advices not going to narrow roads and the author can confirm the feeling of unsafety when walking this specific spot.

One aspect that was noticed in Salway Place entrance area 38 is that edges/roads of that area are of type pedestrian (`highway=pedestrian`) and it's close to a service street (`highway=service`). So a cost will be added to each crime that is near a street that has the mentioned characteristics(See Section 3.6.9 for more details). With these adjustments, the algorithm avoids the narrow road. It chooses a more residential area(Earlham Grove, Keogh Road), then passes through Grove Crescent and finally reaching Great Eastern Road.

Figure 38: Salway Place.
Safe Route, Busyness Points, Safe Route with $wSusRoadCloseToService = 0$, Roads of type service, Salway Place area

Figure 39: Salway Place via Google Maps. It is visible that is quite narrow.

Figure 40: Salway Place corner via Google Maps. This hidden corner can be a spot for malicious actions.

4.4 Discussion

The routes generated on the Experiments section followed the expected behaviour, that is, being different than the fastest route and considering factors such as crime and road characteristics to generate of what might be considered a safe route. It avoided crime(Sections 4.3.1, 4.3.4), it avoided unlit roads(Sections 4.3.2, 4.3.3), prioritised busy roads(section 4.3.1) and avoided suspicious roads as it it can seen in section 4.3.5.

The performance needs to be improved in order to remove the max values done in section 3.6.2. Alternatively research a better method to choose the max values for the cost.

The implementation of this project is not totally robust because there is a lack of automatic testing for the tools and Web Application.

Since the assessment is based on interpretation, there could be arguments of what a route might be considered/feel safe. To decrease this doubts, the police recommendations (Lancashire Constabulary 2022)(Nottinghamshire Police 2022) was used for the creation of this algorithm. A method of enhancing the assessment of whether a route is safe or not, is to generate multiple routes that use different weights and make a survey for residents. They would answer which route seems the safest and improve the algorithm based on their feedback.

5 Conclusion

This article succeeded on the premise of creating an algorithm that considers crime rate and other street characteristics to generate a safe path. Additionally an Web Application was created to demonstrate its usability and potential. It effectively created different routes than popular applications such as Google Maps that only focus on the fastest route.

For this project, only free open source technologies were used, so everyone has access to these tools and data, therefore everyone has the chance of improving on this subject of planning safe pedestrian routes.

This article can also be a starting point for anyone who wants to start researching on subjects that rely on spatial data, since it introduces platforms for geospatial analysis and interactivity.

As for future work:

- Implement unit testing for the tools created and web application.
- Improve the performance of the cost algorithm and implement it on different path-finding algorithms.
- Optimize parameters (Yao et al. 2017).
- Normalize the Risk Score.
- Using different crime density models such as Khalid, Shoaib, Qian, Rui, A. I. Bari, et al. (2018)
- Experiment with different radius for crime analysis($CrimeR$) as pointed out by (Okabe, Satoh, and Sugihara 2009) and adjust the parameters to behave well with such radius.

Acknowledgements

The author wants to thank his tutor, Jia Wang, for guidance on tools, and highlighting the challenges that this project would create.

References

Aziz, Norazah Abd and Raja Mohamad Fairuz R. Mohamad Yusoff (2020). "Density of Route Frequency for Enforcement". In: *Proceedings of the 2020 8th International Conference on Information and Education Technology*. ICIET 2020. Okayama, Japan: Association for Computing Machinery, pp. 247–251. ISBN: 9781450377058. DOI: 10.1145/3395245.3396207. URL: https://doi.org/10.1145/3395245.3396207.

bboxfinder.com contributors (June 2019). *bbox finder*. URL: http://bboxfinder.com/#51.512604,-0.013602,51.551731,0.085248 (visited on 2022-06-30).

Box, G. E. P. and David A. Pierce (1970). "Distribution of Residual Autocorrelations in Autoregressive-Integrated Moving Average Time Series Models". In: *Journal of the American Statistical Association* 65.332, pp. 1509–1526. DOI: 10.1080/01621459.1970.10481180. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1970.10481180. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1970.10481180.

Burke, Edmund K. et al. (Dec. 2013). "Hyper-heuristics: a survey of the state of the art". In: *Journal of the Operational Research Society* 64.12, pp. 1695–1724. ISSN: 1476-9360. DOI: 10.1057/jors.2013.71. URL: https://doi.org/10.1057/jors.2013.71.

Butler, H. et al. (Aug. 2016). *The GeoJSON Format*. DOI: 10.17487/RFC7946. URL: https://www.rfc-editor.org/info/rfc7946 (visited on 2022-06-19).

Carlson, S. C. (June 2022). *Graph Theory*. URL: https://www.britannica.com/topic/graph-theory (visited on 2022-06-19).

City of London Police (June 2022). *Home — City of London Police*. URL: https://www.cityoflondon.police.uk/ (visited on 2022-06-14).

data.police.uk (June 2022a). *About — data.police.uk*. URL: https://data.police.uk/about/ (visited on 2022-06-14).

— (June 2022b). *Data Downloads — data.police.uk*. URL: https://data.police.uk/data/ (visited on 2022-06-14).

Deb, K. et al. (2002). "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE Transactions on Evolutionary Computation* 6.2, pp. 182–197. DOI: 10.1109/4235.996017.

GDAL/OGR contributors (2022a). *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation. DOI: 10.5281/zenodo.5884351. URL: https://gdal.org.

— (June 2022b). *ogr2ogr - GDAL documentation*. URL: https://gdal.org/programs/ogr2ogr.html (visited on 2022-06-19).

Google (June 2022a). *Get information about busy areas from Google Maps*. URL: https://support.google.com/maps/answer/11323117?hl=en (visited on 2022-06-19).

— (June 2022b). *Google Maps*. URL: https://www.google.co.uk/maps/ (visited on 2022-06-14).

Hart, Peter E., Nils J. Nilsson, and Bertram Raphael (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2, pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

Hellerstein, Joseph M. (Aug. 1999). *GiST: A Generalized Search Tree for Secondary Storage*. URL: http://gist.cs.berkeley.edu/gist1.html (visited on 2022-07-14).

Institute for Government (Apr. 2021). *Timeline of UK government coronavirus lockdowns and restrictions*. URL: https://www.instituteforgovernment.org.uk/charts/uk-government-coronavirus-lockdowns (visited on 2022-06-04).

Islam, Khawar and Akhter Raza (2020). "Forecasting Crime Using ARIMA Model". In: *CoRR* abs/2003.08006. arXiv: 2003.08006. URL: https://arxiv.org/abs/2003.08006.

Johnson, Boris (Mar. 2020a). *Prime Minister's statement on coronavirus (COVID-19): 16 March 2020*. URL: https://www.gov.uk/government/speeches/pm-statement-on-coronavirus-16-march-2020#:~:text=And%5C%20even%5C%20if%5C%20you%5C%20don,home%5C%20where%5C%20they%5C%20possibly%5C%20can. (visited on 2022-06-04).

— (Mar. 2020b). *Prime Minister's statement on coronavirus (COVID-19): 23 March 2020*. URL: https://www.gov.uk/government/speeches/pm-address-to-the-nation-on-coronavirus-23-march-2020 (visited on 2022-06-04).

Kenneth Reitz, Python Packaging Authority (June 2022). *Pipenv: Python Dev Workflow for Humans*. URL: https://pipenv.pypa.io/en/latest/ (visited on 2022-06-16).

Khalid, Shoaib, Fariha Shoaib, Tianlu Qian, Yikang Rui, Arezu Bari, et al. (Sept. 2018). "Network Constrained Spatio-Temporal Hotspot Mapping of Crimes in Faisalabad". In: *Applied Spatial Analysis and Policy* 11. DOI: 10.1007/s12061-017-9230-x.

Khalid, Shoaib, Fariha Shoaib, Tianlu Qian, Yikang Rui, Arezu Imran Bari, et al. (Sept. 2018). "Network Constrained Spatio-Temporal Hotspot Mapping of Crimes in Faisalabad". In: *Applied Spatial Analysis and Policy* 11.3, pp. 599–622. ISSN: 1874-4621. DOI: 10.1007/s12061-017-9230-x. URL: https://doi.org/10.1007/s12061-017-9230-x.

Klochovka, Olena et al. (Apr. 2017). "Pedestrian route planning based on an enhanced representation of pedestrian network and probabilistic estimate of signal delays". In: *GISRUK 2017 e-Proceedings*. 63. Conference held from 18-21 April 2017, Manchester, UK. GISRUK, pp. 1–6. URL: http://gala.gre.ac.uk/id/eprint/16575/.

Lancashire Constabulary (June 2022). *Lancashire Constabulary - Lancashire Police - Staying safe when out and about*. URL: https://www.lancashire.police.uk/help-advice/personal-safety/staying-safe-when-out-and-about/ (visited on 2022-06-19).

Metropolitan Police (June 2022). *Home — Metropolitan Police*. URL: https://www.met.police.uk/ (visited on 2022-06-14).

Nottinghamshire Police (June 2022). *Advice guide - Personal safety*. URL: https://www.nottinghamshire.police.uk/advice/prevention/personalsafety (visited on 2022-06-19).

Office for National Statistics (Nov. 22, 2016). *Lower Layer Super Output Area (LSOA) boundaries*. URL: https://data.gov.uk/dataset/fa883558-22fb-4a1a-8529-cffdee47d500/lower-layer-super-output-area-lsoa-boundaries (visited on 2022-06-14).

Office for National Statistics (Jan. 2020). *Crime in England and Wales: Year Ending September 2019*. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/bulletins/crimeinenglandandwales/yearendingdecember2021 (visited on 2022-06-04).

— (Apr. 2022). *Crime in England and Wales: Year Ending December 2021*. URL: https://www.ons.gov.uk/peoplepopulationandcommunity/crimeandjustice/bulletins/crimeinenglandandwales/yearendingdecember2021 (visited on 2022-06-04).

Okabe, Atsuyuki, Toshiaki Satoh, and Kokichi Sugihara (2009). "A kernel density estimation method for networks, its computational method and a GIS-based tool". In: *International Journal of Geographical Information Science* 23.1, pp. 7–32. DOI: 10.1080/13658810802475491. eprint: https://doi.org/10.1080/13658810802475491. URL: https://doi.org/10.1080/13658810802475491.

OpenJS Foundation (2017). *Express - Node.js Web application framework*. URL: https://expressjs.com/ (visited on 2022-06-30).

— (2022). *Node.js*. URL: https://nodejs.org/en/ (visited on 2022-06-30).

OpenStreetMap contributors (May 23, 2021). *Elements - OpenStreetMap Wiki*. URL: https://wiki.openstreetmap.org/wiki/Elements (visited on 2022-06-15).

— (June 2022a). *OpenStreetMap*. URL: https://www.openstreetmap.org (visited on 2022-06-15).

— (June 2022b). *OpenStreetMap*. URL: https://www.openstreetmap.org/about (visited on 2022-06-15).

osm2geojson contributors (June 2022). *GitHub - aspectumapp/osm2geojson - Convert OSM and Overpass JSON to GeoJSON*. URL: https://github.com/aspectumapp/osm2geojson (visited on 2022-06-19).

Overpass API contributors (June 2022a). *Overpass API - OpenStreetMap*. URL: https://wiki.openstreetmap.org/wiki/Overpass_API (visited on 2022-06-15).

— (June 2022b). *Overpass API - OpenStreetMap*. URL: https://wiki.openstreetmap.org/wiki/Overpass_API (visited on 2022-06-15).

— (June 2022c). *The Data Model of OpenStreetMap*. URL: https://dev.overpass-api.de/overpass-doc/en/preface/osm_data_model.html (visited on 2022-06-15).

Pandas (June 2022). *Pandas - Python Data Analysis Library*. URL: https://pandas.pydata.org/ (visited on 2022-06-19).

pgRouting Community (June 2022). *pgRouting Project – Open Source Routing Library*. URL: https://pgrouting.org/ (visited on 2022-06-16).

pgRouting Contributors (Oct. 2021a). *pgr_createTopology - pgRouting Manual (3.1)*. URL: https://docs.pgrouting.org/3.1/en/pgr_createTopology.html (visited on 2022-06-27).

— (Oct. 2021b). *pgr_dijkstra - pgRouting Manual (3.1)*. URL: https://docs.pgrouting.org/3.1/en/pgr_dijkstra.html (visited on 2022-06-27).

Porta, Sergio et al. (2009). "Street Centrality and Densities of Retail and Services in Bologna, Italy". In: *Environment and Planning B: Planning and Design* 36.3, pp. 450–465. DOI: 10.1068/b34098. eprint: https://doi.org/10.1068/b34098. URL: https://doi.org/10.1068/b34098.

POSTGIS (June 2022a). *About PostGIS*. URL: https://postgis.net/ (visited on 2022-06-16).

— (June 2022b). *Geography*. URL: http://postgis.net/workshops/postgis-intro/geography.html (visited on 2022-07-14).

— (June 2022c). *Spatial Indexing - Introduction to PostGIS*. URL: http://postgis.net/workshops/postgis-intro/indexing.html (visited on 2022-07-14).

— (June 2022d). *ST_DWithin*. URL: https://postgis.net/docs/ST_DWithin.html (visited on 2022-06-27).

— (June 2022e). *ST_EndPoint*. URL: https://postgis.net/docs/ST_EndPoint.html (visited on 2022-06-29).

— (June 2022f). *ST_StartPoint*. URL: https://postgis.net/docs/ST_StartPoint.html (visited on 2022-06-29).

Python Software Foundation (June 2022). *Welcome to python.org*. URL: https://www.python.org/ (visited on 2022-06-15).

QGIS contributors (July 2022). *QGIS*. URL: https://www.qgis.org/en/site/ (visited on 2022-07-07).

Ribot, Nicolas (Oct. 2021). *pgr_nodeNetwork - pgRouting Manual 3.1*. URL: https://docs.pgrouting.org/3.1/en/pgr_nodeNetwork.html (visited on 2022-06-27).

Sigaev, Teodor and Oleg Bartunov (June 2002). *GiST Support in PostgreSQL*. URL: http://www.sai.msu.su/~megera/postgres/gist/ (visited on 2022-07-14).

The Matplotlib Development team (June 2022). *Matplotlib - Visualization with Python*. URL: https://matplotlib.org/ (visited on 2022-06-19).

The PostgreSQL Global Development Group (July 2022a). *GiST Support in PostgreSQL*. URL: https://www.postgresql.org/docs/current/gist-intro.html (visited on 2022-07-14).

— (July 2022b). *PostgreSQL: Documentation: 14.8.8. Geometric Types*. URL: https://www.postgresql.org/docs/current/datatype-geometric.html (visited on 2022-07-14).

— (June 2022c). *PostgreSQL: The World's Most Advanced Open Source Relational Databas*. URL: https://www.postgresql.org/ (visited on 2022-06-16).

Tompson, Lisa et al. (2015). "UK open source crime data: accuracy and possibilities for research". In: *Cartography and Geographic Information Science* 42.2, pp. 97–111. DOI: 10.1080/15230406.2014.972456. eprint: https://doi.org/10.1080/15230406.2014.972456. URL: https://doi.org/10.1080/15230406.2014.972456.

U.S. Naval Academy (Aug. 2019). *Approximate Metric Equivalents for Degrees, Minutes, and Seconds*. URL: https://www.usna.edu/Users/oceano/pguth/md

_help/html/approx_equivalents.htm (visited on 2022-06-30).

Utamima, Amalia and Arif Djunaidy (2017). "Be-safe travel, a web-based geographic application to explore safe-route in an area". In: *AIP Conference Proceedings* 1867.1, p. 020023. DOI: 10.1063/1.4994426. eprint: https://aip.scitation.org/doi/pdf/10.1063/1.4994426. URL: https://aip.scitation.org/doi/abs/10.1063/1.4994426.

Vladimir Agafonkin, Leaflet contributors (June 2022). *Leaflet*. URL: https://leafletjs.com/ (visited on 2022-06-30).

Yang, Byungyun (2019). "GIS Crime Mapping to Support Evidence-Based Solutions Provided by Community-Based Organizations". In: *Sustainability* 11.18. ISSN: 2071-1050. DOI: 10.3390/su11184889. URL: https://www.mdpi.com/2071-1050/11/18/4889.

Yao, Yuan et al. (2017). "An efficient learning-based approach to multi-objective route planning in a smart city". In: *2017 IEEE International Conference on Communications (ICC)*, pp. 1–6. DOI: 10.1109/ICC.2017.7997454.

A Appendix

A.1 Overpass API Query for Street Data

Replace {{bbox}} with 51.51260436919461, -0.013602060765980596, 51.55173130930535, 0.08524790329606169 if querying directly to Overpass API.

```
[out:json] [timeout:25];
(
        nwr["highway"="footway"]({{bbox}});
        nwr["highway"="path"]({{bbox}});
        nwr["highway"="bridleway"]({{bbox}});
        nwr["highway"="pedestrian"]({{bbox}});
        nwr["highway"="residential"]({{bbox}});
        nwr[highway][sidewalk~"^(right|left|both|separate)$"]({{bbox}});
        nwr["highway"="service"]({{bbox}});
        nwr["highway"="cycleway"]({{bbox}});
        nwr["highway"="steps"]({{bbox}});
        nwr[highway=unclassified]({{bbox}});
        nwr["footway"]({{bbox}});
        nwr["foot"~"^(yes|designated|permissive|use_sidepath)$"]({{bbox}});

        nwr["highway"~"^(primary|secondary|tertiary|crossing|living_street)$"]["foot"!="no"]({{bbox
            }});

);
(._;>;);
out body;
```

A.2 Database Query to fetch crimes by geography points(Snap Points)

```
SELECT row_number() OVER (ORDER BY longlat_point), longlat_point, count(*) AS count_total
FROM crimes_street GROUP BY longlat_point
```

A.3 Overpass API Query for busyness sources

Replace {{bbox}} with 51.51260436919461, -0.013602060765980596, 51.55173130930535, 0.08524790329606169 if querying directly to Overpass API.

```
[out:json];
(
nwr[amenity~"^(bar|biergarten|cafe|fast_food|food_court|ice_cream|pub|restaurant|college|
    driving_school|kindergarten|language_school|library|toy_library|music_school|school|university|
    police|post_depot|post_office|courthouse|fire_station|arts_centre|casino|cinema|community_centre|
    conference_centre|events_venue|fountain|gambling|planetarium|social_centre|theatre|studio|
    nightclubbaby_hatch|clinic|dentist|doctors|hospital|nursing_home|pharmacy|social_facility|
    veterinarybus_station|bus_station|dentist|doctors|hospital|nursing_home|bicycle_rental|
    boat_rental|car_rental|car_wash|car_wash|vehicle_inspection|charging_station|fuel|taxi|pharmacy|
    social_facility|veterinary|marketplace|place_of_worship)$"]({{bbox}});
nwr[tourism~"^(attraction|hotel|hostel|gallery|museum)$"]({{bbox}});
nwr[public_transport~"^(platform|station)$"]({{bbox}});
nwr[shop]({{bbox}});
nwr[office]({{bbox}});

);
(._;>;);
```

A.4 Query to count crimes by location

```
SELECT row_number() OVER (ORDER BY longlat_point), longlat_point, count(*) AS count_total FROM
    crimes_street GROUP BY longlat_point
```

A.5 Query to count crimes near edge

```
SELECT COUNT(*) FROM crimes_street as cst WHERE ST_DWithin(
(SELECT geom FROM edges_noded WHERE edges_noded.id = 7736), cst.longlat_point, 100);
```

A.6 Query to count average crime per Snap Point

```
SELECT AVG(count_by_geom) FROM
(SELECT COUNT(*) as count_by_geom FROM crimes_street GROUP BY longlat_point) count_by_geom_select
```

A.7 Query to calculate cost

NOTE: This core query is correct, but the syntax is invalid. Every substring that starts with $ is just there for readability. Replace them with integers or PostgreSQL's prepared parameters and it will work.

```
SELECT (geom_length +
        geom_length * $wCrime * (
          SELECT COUNT(*) FROM crimes_street as cst WHERE ST_DWithin(edges_noded.geom, cst.
              longlat_point, 100) LIMIT 20
         ) +
    geom_length * $wNotLit * (NOT is_lit(lit)::boolean)::int +
    geom_length * $wServiceType * (
            CASE WHEN service=''alley'' OR service=''parking_aisle'' THEN 1 ELSE 0 END
        ) +
    geom_length *
          ($wLackBusy - (SELECT (($wLackBusy / 15) * (SELECT COUNT(*) as records FROM (
                    SELECT busy_points.id FROM busy_sources_points as busy_points
                    INNER JOIN edges_noded as edges_noded_temp
                            ON ST_DWithin(edges_noded_temp.geom, busy_points.geom, (15.0 / 100000))
                    WHERE edges_noded_temp.id = edges_noded.id LIMIT 15
          ) busy_points_around )) as Weight)
        ) +
    geom_length * 0.03 *
      (CASE WHEN (sidewalk=''separate'' OR sidewalk_left=''separate'' OR sidewalk_right=''separate'')
      THEN 1 ELSE 0 END) +
    geom_length * $wCurrCrimes * (
          SELECT COUNT(*) FROM current_crimes as current_cst WHERE ST_DWithin(edges_noded.geom,
              current_cst.geom, 150) LIMIT 5
        ) +
    geom_length * 2 * (CASE WHEN access=''private'' THEN 1 ELSE 0 END) +
    (CASE WHEN highway IN (''pedestrian'') AND
            (
              SELECT COUNT(*) FROM edges_noded as edges_noded_temp WHERE edges_noded_temp.highway=''
                  service'' AND
              (ST_DWithin(edges_noded_temp.geom, ST_StartPoint(edges_noded.geom)::geometry, 35) OR
                  ST_DWithin(edges_noded_temp.geom, ST_EndPoint(edges_noded.geom)::geometry, 35))
                  LIMIT 1
            ) > 0
          THEN
            geom_length * (SELECT COUNT(*) FROM crimes_street as cst WHERE ST_DWithin(edges_noded.
                geom, cst.longlat_point, 25)) * $wSusRoadCloseToService
          ELSE 0 END
    ) +
    (CASE WHEN (service IS NULL OR service != ''parking_aisle'') AND
      (SELECT COUNT(*) FROM edges_noded as edges_noded_temp WHERE ST_DWithin(edges_noded_temp.geom,
          ST_Centroid(edges_noded.geom), 20.0 / 100000) AND service=''parking_aisle'' LIMIT 1) > 0
      THEN (geom_length * $wServiceType)
      ELSE 0 END
    )


    ) as cost FROM edges_noded
```

A.8 Query to generate safe route

NOTE: This core query is correct, but the syntax is invalid. Every substring that starts with $ is just there for readability. Replace them with numbers or PostgreSQL's prepared parameters and it will work.

```
SELECT seq,path_seq,node,edge,cost,agg_cost,
edges_noded.id as edge_id, is_lit(edges_noded.lit) as is_lit, edges_noded.geom as geom
FROM pgr_dijkstra('SELECT id, source, target,
      (geom_length +
        geom_length * $wCrime * (
          SELECT COUNT(*) FROM crimes_street as cst WHERE ST_DWithin(edges_noded.geom, cst.
              longlat_point, 100) LIMIT 20
         ) +
    geom_length * $wNotLit * (NOT is_lit(lit)::boolean)::int +
    geom_length * $wServiceType * (
            CASE WHEN service=''alley'' OR service=''parking_aisle'' THEN 1 ELSE 0 END
        ) +
    geom_length *
          ($wLackBusy - (SELECT (($wLackBusy / 15) * (SELECT COUNT(*) as records FROM (
                    SELECT busy_points.id FROM busy_sources_points as busy_points
                    INNER JOIN edges_noded as edges_noded_temp
```

```
                        ON ST_DWithin(edges_noded_temp.geom, busy_points.geom, (15.0 / 100000))
                    WHERE edges_noded_temp.id = edges_noded.id LIMIT 15
          ) busy_points_around )) as Weight)
        ) +
    geom_length * 0.03 *
      (CASE WHEN (sidewalk=''separate'' OR sidewalk_left=''separate'' OR sidewalk_right=''separate'')
      THEN 1 ELSE 0 END) +
    geom_length * $wCurrCrimes * (
        SELECT COUNT(*) FROM current_crimes as current_cst WHERE ST_DWithin(edges_noded.geom,
            current_cst.geom, 150) LIMIT 5
      ) +
    geom_length * 2 * (CASE WHEN access=''private'' THEN 1 ELSE 0 END) +
    (CASE WHEN highway IN (''pedestrian'') AND
            (
              SELECT COUNT(*) FROM edges_noded as edges_noded_temp WHERE edges_noded_temp.highway=''
                  service'' AND
              (ST_DWithin(edges_noded_temp.geom, ST_StartPoint(edges_noded.geom)::geometry, 35) OR
                  ST_DWithin(edges_noded_temp.geom, ST_EndPoint(edges_noded.geom)::geometry, 35))
                  LIMIT 1
            ) > 0
          THEN
            geom_length * (SELECT COUNT(*) FROM crimes_street as cst WHERE ST_DWithin(edges_noded.
                geom, cst.longlat_point, 25)) * $wSusRoadCloseToService
          ELSE 0 END
    ) +
    (CASE WHEN (service IS NULL OR service != ''parking_aisle'') AND
      (SELECT COUNT(*) FROM edges_noded as edges_noded_temp WHERE ST_DWithin(edges_noded_temp.geom,
          ST_Centroid(edges_noded.geom), 20.0 / 100000) AND service=''parking_aisle'' LIMIT 1) > 0
      THEN (geom_length * $wServiceType)
      ELSE 0 END
    )


    ) as cost FROM edges_noded
', $startVertex, $endVertex, false) as route
LEFT JOIN edges_noded ON edges_noded.id = route.edge
LEFT JOIN edges_noded_vertices_pgr ON edges_noded_vertices_pgr.id = node
```