CTEC3451 – Final Year Project

Final Deliverable

How much data can an application gather from an individual and how can this be a security risk to an individual?

Alexander Thomas Hulme

BSc Cyber Security

P2508338

Supervisor

Dr Leandros Maglaras

School of Computer Science and Informatics

De Montfort University

Contents

Acknowledgements

I would like to thank my project supervisor Leandros Maglaras for his support during this project and for allowing me to expand on my ideas to a point in which I could discuss this with others. As well as Leandros, I would like to thank Ioannis Vourgidis and Leo Tisljaric for taking time out of their PhD studies to help me in my project by sharing their knowledge and experience within this area as without this support I would have been stuck at the concept of this project.

I would like to thank Drew Gamble for helping me with dev-ops and problem solving by sharing his knowledge in Python to help further my knowledge and understanding while coding. As well as being a fresh set of eyes to look at the code when I was burning out. I really appreciate you and you have done an amazing job as a Course Rep, especially during the pandemic dealing with all sorts of issues in your spare time.

I would also like to thank my friends and family for supporting me through the stressful period of my third year during the COVID-19 pandemic, as well as those on my course who kept an eye out for myself and other students who may have struggled without any interaction with other humans. Special mentions go to Tyler Arthur and Rory Kester who happily proofread my work over the past three years to ensure that what I had wrote was legible. Special thanks to Megan John, who read this entire development project to ensure it was written in a legible format and not full of grammar mistakes.

My thanks go out to the East Midlands University Officer Training Corps, 53 Intelligence Corps and 38 Platoon of CCS203. Members from all three of these groups have supported me throughout my military reserve career and allowed for me to discuss topics involved in my degree with them, this allowed for me to stay engaged with the subject and expand my horizons onto deciding to study a Master’s Degree in Cyberpsychology.

Finally, I would like to thank my housemates, Andre Sobol, Oliver Shalvey-Hill, Catriona Fryer and Chloe Smith, who have lived with me for the past two years and listened to me when I hit a roadblock on a project and given me other angles to view a problem from, as well as some ideas which I wouldn’t have thought of implementing in my work over my time living with them.

Abstract

This report will cover a prototype of a mapping system that has been developed using Python to create a script to analyse a dataset that has been obtained from open sources and later visualised using a map from OpenStreetMaps. The idea behind my project is to provide insight into how an individual's day to day life could be tracked and traced by a malicious application by a malicious third party, particularly this is aimed at military and law enforcement personnel as well as medical workers in some countries. The reasoning behind this is due to people working in these areas have a higher chance of being attacked, whether this is by international terrorist organisations or internal groups which disagree with their job (particularly seen after the death of George Floyd in 2020.) This application should show users where they frequent from data sourced from OpenStreetMaps and then rank how often they are there in a set timeframe in a database list and give a threat rating.

This paper follows on from my previous paper and will follow the development and implementation of the ideas highlighted in said paper.

Introduction

In modern day society we are reliant on technology, to the point that we scaled down a computer and combined it with a phone to create a smart phone. As a planet we are more connected than ever, applications are in an abundance, whether this is using maps to get to the grocery store or your music application using your location to find gigs of your favourite artists. But the more open society becomes to technology, there will always be those who use this as a way of exploiting applications to their advantage or some applications which innocently put people at risk. As mentioned in the previous paper, Strava was cause for concern when its innocent feature of heat mapping accidentally revealed locations of military bases and forward operating bases in the Middle East and Africa (Hern, 2018).

The risk of applications that track locations is the fact that this information in some jobs can be in breach of OPSEC and PERSEC, which is operational security and personal security. Operational security when breached can put hundreds or even thousands of lives at risks in an operational combat theatre (such as Afghanistan, Iraq or even on peacekeeping missions,) while personal security when breached can put the individual at risk as well as those immediately around them such as family, friends or work colleagues. For those working in Law Enforcement, Military and in some medical backgrounds, having watertight personal security can be the difference between life and death due to the ever-growing threat of both foreign and internal organisations that might not agree with their job.

The code for this project is viewable via the following:

https://github.com/JagerScouser/Alex_Hulme_FYP_v2

Motivation

During my time as a Cyber Security student at De Montfort University, I have been exposed to a wide range of topics in the cyber world, some I could apply to my day-to-day life and some I wondered if I would use these skills again. I became interested in ethics and law while I was studying, and this resulted in me being fixated on the right to privacy and what companies can do with users data or what data can be collected without the user's knowledge.

A user in an online forum said "if you are not paying for it, you're not the customer; you're the product being sold." (blue_beetle, 2010) This phrase has been reused so many times to say that your data is always what a company is interested in, so they can sell it off to service providers to target you with advertisements. This was always an interest to me, as a user of many social media sites for free, I did notice a lot of what I was looking at online started becoming targeted at myself.

In Autumn 2020 I was away with the military at the Royal Military Academy Sandhurst, as part of the training we had to learn about the importance of OPSEC (Operational Security) and PERSEC (Personal Security.) I learnt that a lot of applications that I use frequently like Strava and Tinder would be a giveaway to my location if I forgot to turn them off before deployment or even used it while on deployment. I also learnt of how these applications caused security breaches by accident by soldiers on the ground due to the heat maps being active in the deserts of the Middle East. As well as heat maps, most military units have a running or fitness group inside the app and people outside this group can see who's in the group which was not ideal as it could reveal full units operating in certain areas.

After finishing my time at the Academy, I was interested in how PERSEC could apply to the day-to-day life of individuals in different sectors of work. What were the risks? Was it exclusively just military personnel that were at risk or was it everyone regardless of the job? These questions helped me form my motivation behind my final year project.

The Problem and Objectives

Many applications are released on the market for users to download on their mobile devices both on the Apple store and the Google play store. Many of these apps will innocently use your location to target you for advertisements that are relevant to your area or so you can use their services, such as uber which needs to know your location so a driver can reach you (Lloyd, 2018). With so many innocent apps collecting information on the user, it is entirely possible to assume that there will be some unsavoury applications out there which aim to use the information for other malicious purposes, ranging from stalking to physical attacks (Akinode, 2011).

The objective at the end of this development project is to be able to highlight to a specific user group, how their regular activities can become something that risks their personal security and potentially their lives.

The groups targeted for this project are:

- Military personnel
- Law enforcement officers
- Medical Workers
- Politicians

All these groups are targeted for different reasons and as a result they need to have a tight level of personal security. Military personnel are prime targets for extremist groups, both domestic such as the Irish Republican Army (IRA) and multi-national threats such as Al-Qaeda and the Islamic State of Iraq and the Levant (ISIL/Daesh) which all have motives ranging from uniting “occupied” lands to conducting an anti-west Jihad (Home Office, 2020). When it comes down to warfare, one of the most crucial information a commander of armed forces can have is up-to-date location of the enemy forces in relation to their own so they can mount offensive operations or prepare courses of action in case of attack from the enemy (Baijal, 2009). So, it is vital for commanders to protect their forces from the enemy by operating a strict Operational Security (OPSEC) and Personal Security (PERSEC) to deny enemy combatants from having access to the “most crucial” information on soldiers whether they are on the battlefield or in the barracks. As seen with the murder of Fusilier Lee Rigby, his attackers had been seen monitoring the area around his barracks looking for a soldier to attack to achieve their goal of a Jihad.

Law enforcement officers also see the same threat from domestic terrorist groups, mainly this is seen in the Police Service of Northern Ireland (PSNI) where the terrorist threat level has been severe since 2010 (MI5, 2021). This is due to the religious tensions in the region between Catholics and Protestants after the Troubles and Good Friday Agreement. Police are also at risk of international terrorism as seen with the attacks at London Bridge and Westminster in 2017, where a Police Officer was fatally stabbed and multiple injured from the two attacks. More recently, police have seen violence against them in response to the murder of George Floyd in the United States of America. The protests became a worldwide movement and was seen by some as more than just a justice movement and more of an “anti-cop” movement, police officers within the UK were also assaulted and coughed on (during the COVID-19 pandemic) during these protests (Clifton, 2020).

Medical workers are targets for domestic violence from groups, some groups will target medical workers for providing a treatment or a service which goes against their own morals or religion's standpoint. In South Africa, even paramedics and junior doctors are subject to attacks to a point where paramedics are having to wear bulletproof vests to carry on their work due to the frequency of incidences (ENCA, 2019) (IOL, 2020). This is due to medical care in some countries not being up to the same standard as we see in the United Kingdom and medical supplies catch a high price in organised crime rings. The risk is that those who are known as medical workers with access to these supplies could potentially be monitored by these crime rings and forced through intimidation or physical attacks to get items for the crime rings or have individuals harass them on their daily routine for offering certain treatments.

Finally, in the situations involving politicians, who are always in the public eye and due to the multiple party system that is active in the United Kingdom, there will always be political divides between supporters of one party and the politicians of the opposing party. Normally this is seen between Labour supporters not agreeing with the Conservative politicians in the United Kingdom and vice versa. Whether this be because they supported the war effort in Iraq, as seen in the attack on Stephen Timms or for purely political motives as seen with the murder of Jo Cox (The Guardian, 2016).

Overall, with the proliferation of GPS technology has created risks in ways that individuals could be tracked down by their smartphone or other device down to the meter. It is only natural for individuals to value their privacy and become educated about how they could be predictable and how this could be used for malicious purposes. (Katina Michael, 2006) (Katina Michael, 2013)

Background Research

To get an understanding what was already being developed by individuals, not companies and then decide how to collect GPS data and how to access that data to implement it into a program. After speaking with Ioannis Vourgidis, a PhD student who was making a similar project that was predicting car movements to minimise risk and injuries. He was using a spatial database via PostGres and PostGIS, then visualising this with OpenStreetMaps and interacting with the spatial database via an SQL Client. This concept was used to form the initial base for the project, but this was later scrapped.

I then looked at my own skill set, what I had experience in and what I would be able to apply it into, this happened to be Python due to me studying it from college through to my second year cryptography project. I found a GitHub project by user DebasmitaGhose, the project was for determining modes of transportation using mobile phones using Python and GPS datasets. The project was done as part of a Development Project at the Manipal Institute for Technology and involved many features which caught my interest and could be developed on to aid me in my own project, this included using Python to extract and analyse GPS data from a dataset and involved using a dataset that was from mobile devices. All this current research could be used as a base for me to further develop it into an application that can analyse just the GPS Data and visualise this so the user can see a threat index which the application should produce.

When I initially spoke with Ioannis, he suggested the use of Open Street Maps alongside some database software such as PostGres and PostGIS which the information stored could visualised on a map. I scrapped this idea, but when I conducted further research, I saw an article which showed me how to create a visualisation of GPS data from a dataset in Python (Tisljaric, n.d.) The article shows that it is possible to map out a dataset on a static map, this means that I would have to load the map from Open Street Maps of the area the individual has been in and superimpose the location information over the map. It does this via converting the image pixels to coordinates based off the static map's width and height. As it is a static image it wouldn't be interactive like Google or Apple Maps, but it would show the information required, the main issue would be that the larger the distance covered by the individual the larger a map would be needed and result in loss of detail. For example, if someone's life revolves around being in Leicester and they must drive to Manchester for the day, they would have location data that is outside of the map of Leicester so one would have to be scaled up to accommodate the larger sample area.

Project Specification

Functional Requirements

User Interaction

As the application being developed is to show the user where they are potentially at risk when conducting their day to day lives, the application needs to be able to visualise this data for them to see. It is no use to them being show a list of GPS coordinates as they are unable to relate to this and put it into where exactly they have been over a period of time. Without visualising the data for the user, they would be unable to see how their daily routine is predictable to a point of putting them at risk.

As an addition to the visualisation on a map, the plan is to try create a breakdown of their key locations they have visited. This would be more inclusive of those users who have low spec technology or those who might not see how their day-to-day life is predictable, this would give a more professional breakdown of information rather than just giving a map or a list of key locations by itself.

The user will be able to interact and navigate through the application via a number of buttons. The use of buttons also minimises the chance of user error by a small margin, as well as makes it less demanding on the user to try find file locations manually rather than explorer which they would have used in the past.

User Interface

As previously discussed in my first deliverable, the requirements for the user interface are to be simplistic. The main aim of it would be so that the user can easily access the three parts to the application; the login; the threat analysis screen; and the deletion.

The aim is to keep the interface simplistic to allow for users to load the application with their data, and use it unrestricted and without hinderance, even when using low spec computers or a mobile device. This means it does not need a flashy or modern looking interface, just something simple that gets the job done, something you would more likely see on a Windows XP machine than a modern Windows 10 machine. This would be to accommodate future users in areas that isn’t as prolific as Western Nations. The initial focus was on using and collecting dataset that had been collected for this specific project so the focus would be around the United Kingdom, but ultimately this is dictated by where the dataset collects its data from.

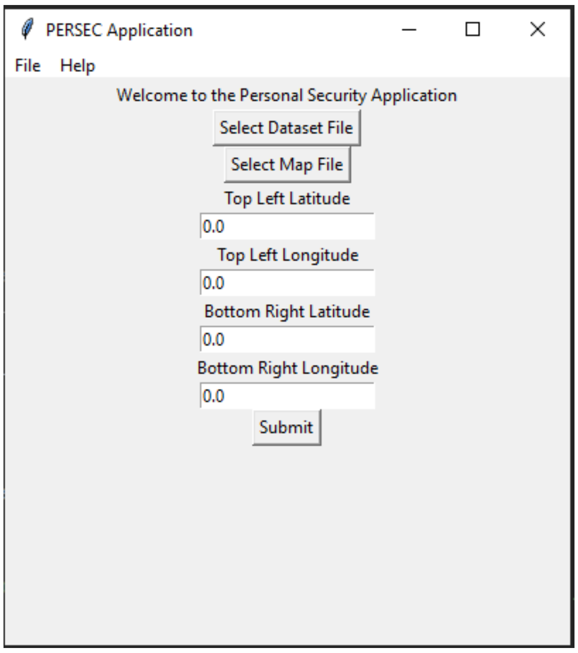
By a simplistic design, it means by using neutral colours that are easy to read and aren’t taxing on the eyes as to include everyone. For example, 8% of men would struggle to see red text on a green background (Colour Blind Awareness, n.d.), ideally the aim is to avoid excluding any users when making the user interface.

System Design

User Interface

The Graphical User Interface (GUI) is the front end of the application, it is what the user will see and interact with. Making this is vital to allow the user to input their data and receive data back with their location plotted on a map.

The initial concept planned on was a simplistic looking model, nothing compared to the likes of a mobile application like Strava or the NHS COVID-19 application, both of which have a simplistic but at the same time graphically and technically demanding. This is due to the fact both use either animation, changing colour schemes and other live interaction features which would be hard to implement without limiting it on what hardware could run it without any issues.


PERSEC Application

File Help

Welcome to the Personal Security Application

Select Dataset File

Select Map File

Top Left Latitude

0.0

Top Left Longitude

0.0

Bottom Right Latitude

0.0

Bottom Right Longitude

0.0

Submit

Figure 1 GUI Initial Layout

For the development of the GUI, PyCharm IDE was used to code the Python script into, this allowed for any errors to be highlighted without having to wait till the point of execution.

Creating the GUI, using the tkinter tool in python and then created an object which would house the GUI, this was named ‘root’ and would be used throughout the development of the GUI to implement features to enhance user interaction.

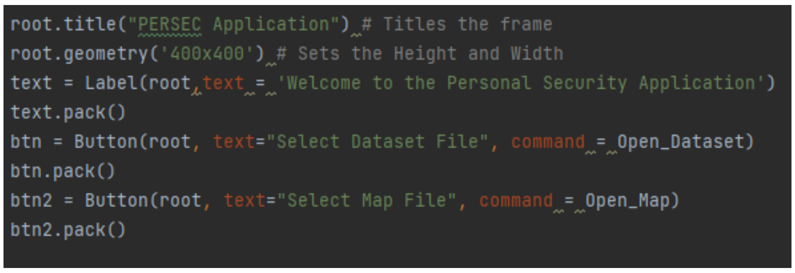

```
root.title("PERSEC Application") # Titles the frame
root.geometry('400x400') # Sets the Height and Width
text = Label(root,text = 'Welcome to the Personal Security Application')
text.pack()
btn = Button(root, text="Select Dataset File", command = Open_Dataset)
btn.pack()
btn2 = Button(root, text="Select Map File", command = Open_Map)
btn2.pack()
```

Figure 3 Creating the Application Window Code

Then began visualising the GUI from the Python code by using the geometry tool to create a 400-pixel by 400-pixel box. Once this line was executed it would create a square pop up that the user could interact with the application in. Once this was working, it was time to make the application look more interactive and user friendly, a title was added to the root which named the program at the top as well as in task manager and the bottom navigation bar on windows. Two buttons were implemented, both would allow the user to open the relevant files so that they could be passed into the back end of the application, these buttons for the initial part were on top of each other so that they could be found easily. The buttons when clicked on by the user would call one of two functions ‘Open_Dataset’ or ‘Open_Map’ relevant to which button was pressed by the user, this would open up a file explorer so the user could navigate and find both the relevant dataset and

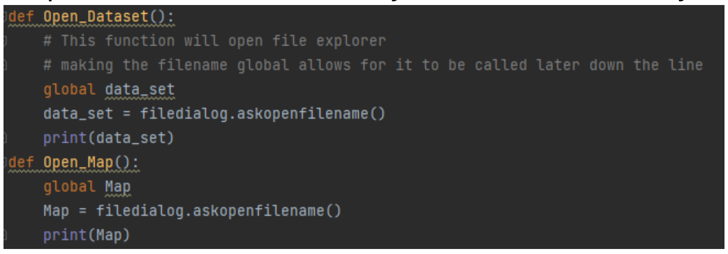
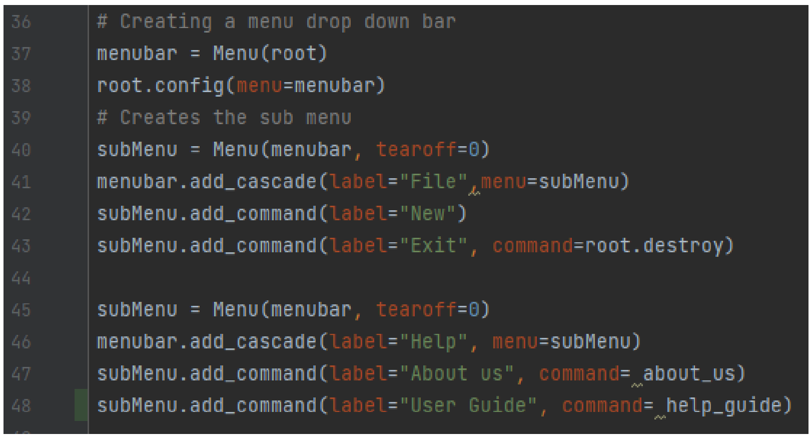

```
def Open_Dataset():
    # This function will open file explorer
    # making the filename global allows for it to be called later down the line
    global data_set
    data_set = filedialog.askopenfilename()
    print(data_set)
def Open_Map():
    global Map
    Map = filedialog.askopenfilename()
    print(Map)
```

Figure 2 Button Commands Code

the map which could be uploaded so data could be plotted on it.

All the buttons and labels had to be packed, this allowed for it to be placed within the GUI geometry window which had been created, without doing this the code would execute but the buttons and labels would not appear on the GUI and not give any errors. As well as ensuring that the relevant features were packed within the 'root.mainloop()', this is due to the fact the initial GUI would appear for a fraction of a second and disappear as it is executed and the infinite loop keeps the program window open unless it is terminated by the user exiting the program or achieving its purpose. In this case the program window would only terminate by the user's choosing.

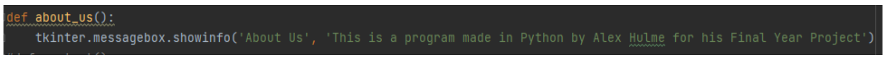


```python
# Creating a menu drop down bar
menubar = Menu(root)
root.config(menu=menubar)
# Creates the sub menu
subMenu = Menu(menubar, tearoff=0)
menubar.add_cascade(label="File",menu=subMenu)
subMenu.add_command(label="New")
subMenu.add_command(label="Exit", command=root.destroy)

subMenu = Menu(menubar, tearoff=0)
menubar.add_cascade(label="Help", menu=subMenu)
subMenu.add_command(label="About us", command= about_us)
subMenu.add_command(label="User Guide", command= help_guide)
```

Figure 4 Sub Menu Code

Later a menu bar was implemented to the program, this allows me to embed some extra features within the program so that it does not occupy the GUI's main window and so the user doesn't accidentally click on a feature like 'exit' or 'new' when inputting data and having to restart inputting data.

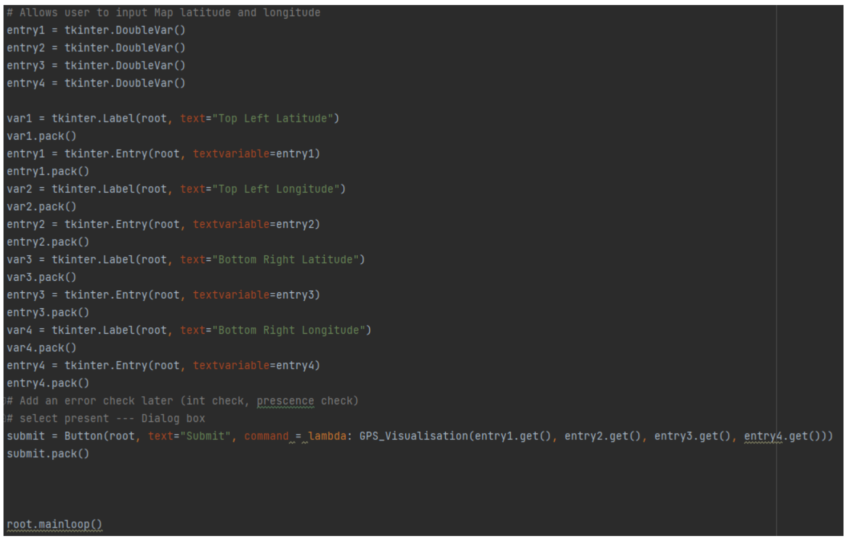
By using the cascade feature so that the menu would drop down and allow multiple options under suitable headers such as 'File' hosting both 'New' and 'Exit' and 'Help' hosting 'User Guide' and 'About Us.' All of these when clicked on would have their own purpose, exit would use a destroy command that would terminate the root and thus the program would shut down without saving data. About Us would call a command that executed a message box that shows information about the project and its purpose.



```python
def about_us():
    tkinter.messagebox.showinfo('About Us', 'This is a program made in Python by Alex Hulme for his Final Year Project')
```

Figure 5 About Us Command Code

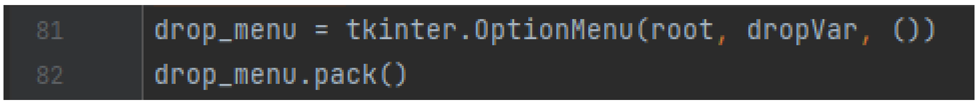
The final implementation was the user inputs of the latitude and longitude of two points, as well as a submit button. The top left of the map and the bottom right of the map so that the back end of the program can work out where to plot on the map.



```python
# Allows user to input Map latitude and longitude
entry1 = tkinter.DoubleVar()
entry2 = tkinter.DoubleVar()
entry3 = tkinter.DoubleVar()
entry4 = tkinter.DoubleVar()

var1 = tkinter.Label(root, text="Top Left Latitude")
var1.pack()
entry1 = tkinter.Entry(root, textvariable=entry1)
entry1.pack()
var2 = tkinter.Label(root, text="Top Left Longitude")
var2.pack()
entry2 = tkinter.Entry(root, textvariable=entry2)
entry2.pack()
var3 = tkinter.Label(root, text="Bottom Right Latitude")
var3.pack()
entry3 = tkinter.Entry(root, textvariable=entry3)
entry3.pack()
var4 = tkinter.Label(root, text="Bottom Right Longitude")
var4.pack()
entry4 = tkinter.Entry(root, textvariable=entry4)
entry4.pack()
# Add an error check later (int check, prescence check)
# select present --- Dialog box
submit = Button(root, text="Submit", command = lambda: GPS_Visualisation(entry1.get(), entry2.get(), entry3.get(), entry4.get()))
submit.pack()



root.mainloop()
```

Figure 6 User Input and storage code

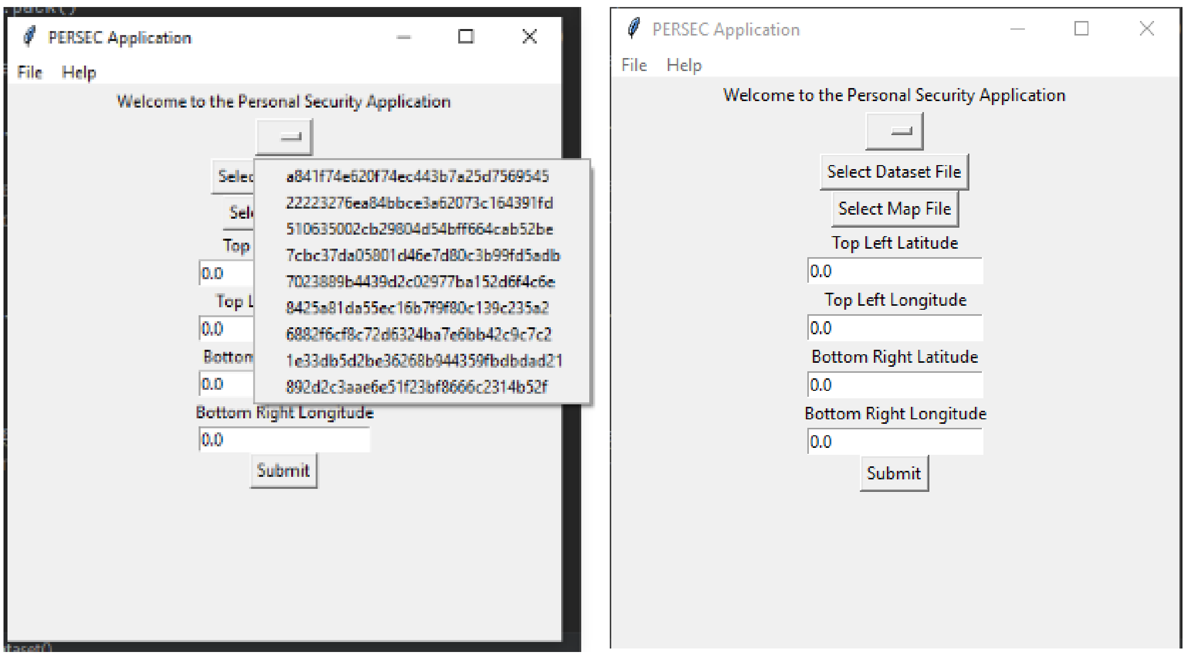
As the program requires two latitudes and two longitude coordinates, it required four input variables that the user would input into the system. For the sake of testing and ensuring they worked a label was created which was named 'var' and had a number from 1 to 4. Each would then carry an 'entry' with a number from 1 to 4, corresponding to which var label they belonged to, this would allow for the user to have a text box in which they could input their coordinates to. Once the user inputs their data into the relevant boxes then they would click the ‘Submit’ button and this would call the ‘GPS_Visualisation’ code in the back end of the application so that it can begin plotting onto the map. The ‘lambda’ was used to allow it to pass multiple entries as an argument so that the program would run once the button was pressed, as due to the infinite loop to keep the GUI open it would run the submit button script and cause errors within the program.



```
drop_menu = tkinter.OptionMenu(root, dropVar, ())
drop_menu.pack()
```

Figure 7 Option Menu Code

After ensuring the code would execute and allow for the program to plot correctly, a dropdown list was implemented. This would allow the user to be able to select an iPhoneUID so the program would plot just the data relevant to that which has been selected. Using the tkinter OptionMenu feature, the program was able to house the values obtained by the dropVar and set it so the location was in the root as seen with the rest of the features of the GUI.



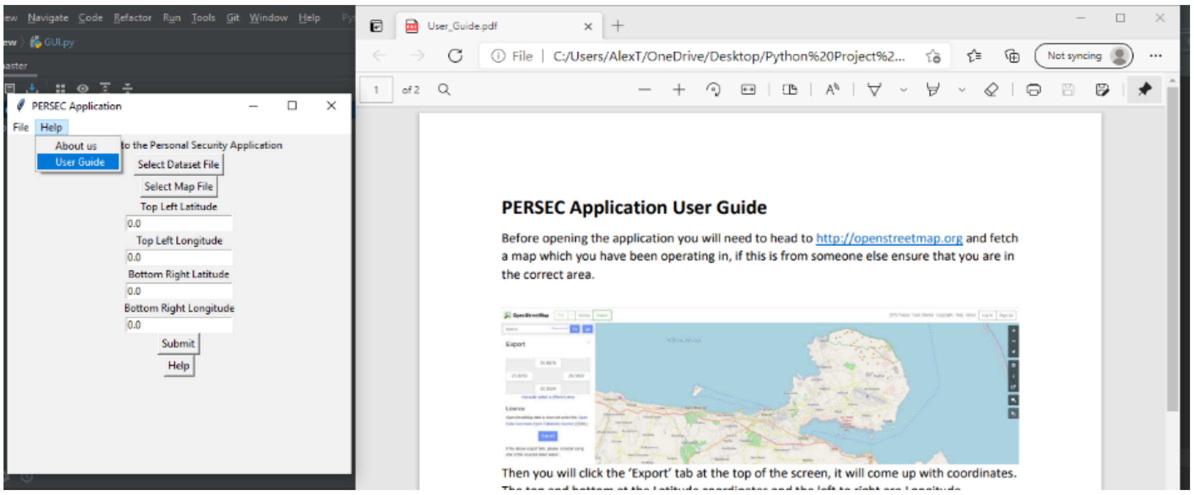
```
a841f74e620f74ec443b7a25d7569545
22223276ea84bbce3a62073c164391fd
510635002cb29804d54bff664cab52be
7cbc37da05801d46e7d80c3b99fd5adb
7023889b4439d2c02977ba152d6f4c6e
8425a81da55ec16b7f9f80c139c235a2
6882f6cf8c72d6324ba7e6bb42c9c7c2
1e33db5d2be36268b944359fbdbdad21
892d2c3aae6e51f23bf8666c2314b52f
```

0.0
0.0
0.0
0.0

Figure 8 Drop Down Menu and Updated GUI

Non-Essential Add-ons

Previously a sub menu was created which would cascade from the top bar when highlighted, as seen in many other applications online. A help guide was added to the project, this was to help the user-friendly side of the project so that the user would be able to understand the project and how to use it to its fullest abilities.



PERSEC Application User Guide

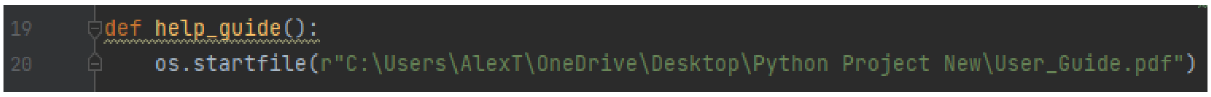
Before opening the application you will need to head to http://openstreetmap.org and fetch a map which you have been operating in, if this is from someone else ensure that you are in the correct area.

Then you will click the 'Export' tab at the top of the screen, it will come up with coordinates.

Figure 9 User Guide Help Tab accessed from the GUI

This would open a PDF which had been created and would show the user how to get their own map from Open Street Maps and how to obtain the coordinates they need to input into GUI so that it would function as intended.

This simply ran by using the following:



```
def help_guide():
    os.startfile(r"C:\Users\AlexT\OneDrive\Desktop\Python Project New\User_Guide.pdf")
```

Figure 10 User Guide Command

As seen in figure 4, when the User Guide label from the cascade was clicked on it would run the command named ‘help_guide’. This command would then use the os.startfile and read feature (‘r’) to read the PDF file that was stored in the location set.

System Implementation

Software Development

Initial Set Up

Initially, it had planned to use PostGres, PostGIS and BeeKeeper Studio to interact with a spatial database and allow for me to create a program which analysed. Although this resulted in many errors while installing PostGIS extension from PostGres which meant that the database unable to use Beekeeper Studio to communicate with the database, thus unable to implement any of my original plans into plan. This idea was then halted while another course of action was planned.

After conducting some research on other applications that interacted with datasets via Python, work began on making an initial static map to upload a dataset to, similar to a previously made software by Github user tisljaricleo. The reasoning being, if a static map could work then data can be uploaded and visualised on the map and from there open the door to further development where automation of the software could be implemented based on where the dataset coordinates were from.

Dataset Acquisition

Initially the focus was to use a dataset from the United Kingdom, this would have involved either having to source a dataset or create a new dataset by gathering it via willing volunteers but this would have been a difficult task and due to COVID-19, due to being unable to meet the individuals in person to collect their data and rely on them uploading it correctly as well as it potentially posing some legal and ethical issues. This project is dictated by the dataset's origin to where the data is visualised, there would be no point in using a dataset in Tokyo, Japan if it was being pitched at a UK audience. Instead the program will use any dataset while prototyping this entire project and become more versatile for users across different regions rather than locking it to one specific region.

The datasets required had to fit a certain format, they had to be open source data so that they would be freely available use the data and had to be from a mobile phone's GPS system. Initially a dataset was sourced through the Google Cloud Platform named 'openmobiledata_public' (Google Cloud Platform, 2014) which met the criteria of being open source but upon inspection there was no data history or how this data was procured. Was this data collected through legal and ethical routes or was the data collected illegally? The only data that was given was it was procured from the United States and was accessible to those who are signed in via a Google Account. Due to the limited data behind the dataset, it was not selected for use in the project.

Another site was used called CRAWDAD, a website which is an open community set up by Dartmouth College, New Hampshire, United States. The site offers datasets from multiple sources and has a filter that allows for a specific search for just GPS related datasets that have been uploaded by the community. This community ranges from individuals doing private research with these datasets to educational institutes such as the Royal Institute for Technology in Stockholm conducting research on their campuses.

Another dataset found was mentioned in the 'Determination of Transportation Modes Using Mobile Phones' paper (Ghose, 2017), this dataset was the 'Geolife GPS trajectory dataset' which was collected by Microsoft Research Asia. The dataset was collected over 4 years with 178 individuals involved in the study and included the data on the outdoor movement of individuals, from going to work and home through to outdoor leisure activities. (Microsoft Research Asia, 2011)

Visualisation of Information

After reading through both ‘Determination of Transportation Modes using Mobile Phones’ (Ghose, 2017) and ‘GPS-Visualisation-Python’ (Tisljaric, n.d.), this gave a brief idea of how to visualise data on a map. Mainly from the user Tisljaric on Github, the software he had created would import the dataset and map it out for the user as seen in Strava or Google Timelines but would not include speeds or the time spent stationary in that area, something that is being planned on visualising later down the line. The software works by superimposing the dataset in a line on a map that relates to the coordinates, this map is a static map as the user would have to upload the map based on their dataset.

The objective is to automate this once the original software works in its intended purposes and so that users can use it regardless of where they are in the world, whether the dataset they have is for Austria or Australia.

By reading the code from ‘GPS-Visualisation-Python’ and gaining an understanding of how it was working, modification began with adding in features that were missing from the original code where deemed fit, as well as providing more in depth comments on the code so that it would be workable by other users so they can modify it. The program works in two parts a ‘Main’ and a ‘GUI’ python script, the ‘GUI’ being where the user will input the file name for the map they have downloaded from open street maps and the coordinates of the map (the top left to the bottom right.) By inputting the coordinates of the map, it plots the dataset accurately which is needed to highlight security issues.

Main

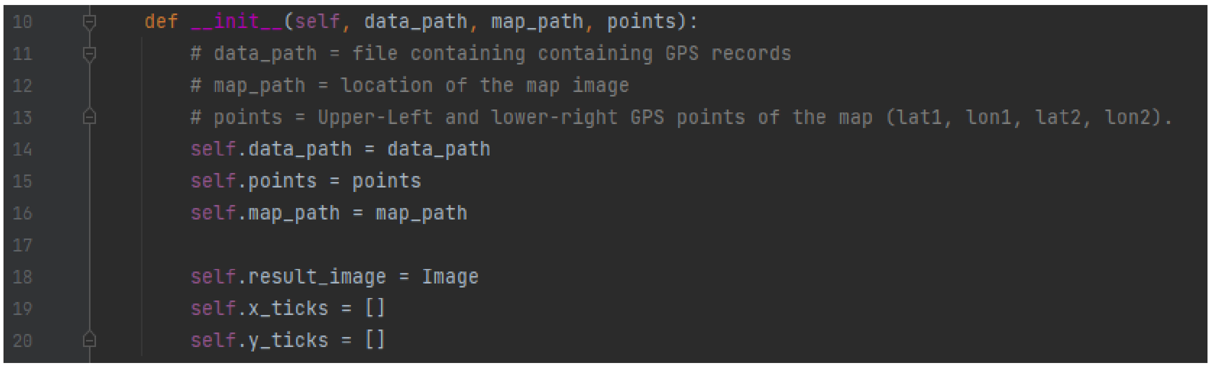
The ‘Main’ project will house all the functions which are involved in creating the actual visualisation so that it could be called by the GPS_Visualisation class once the submit button is pressed on the GUI. As the program is coded in Python, there are several tools and library repositories that allow for the expansion of what python is able to do as a standalone software, thus being able to tailor it to the project’s needs.

A tool named Pandas, this is an open source data analysis and manipulation tool which is built around Python. It will allow for the program to conduct operations at manipulating data in numerical tables or time series, such as the data sets which will be uploaded into the program via the ‘GPS_Visualisation’ class. NumPy, which is a numerical python library that contains arrays and matrix data, this also goes hand in hand with pandas was also imported. This is because Pandas contains objects which rely heavily on NumPy objects.

The last two tools that have been imported are Matplotlib.pyplot which is key for visualisation in python, whether this is from graphs, models or in our case plotting the GPS coordinates from the dataset. This tool will cooperate well with PIL, this is Python Imaging Library. PIL is able to load an image and is able to set its height and width in pixels so that it can only operate in a set area, this is useful as it enables the program to use a tuple and mathematical conversion to convert the dataset coordinates to pixel coordinates on the image or vice versa.

__init__

This function is contained within the GPSVis Class, this is so it can be called by the GPS_Visualisation class.



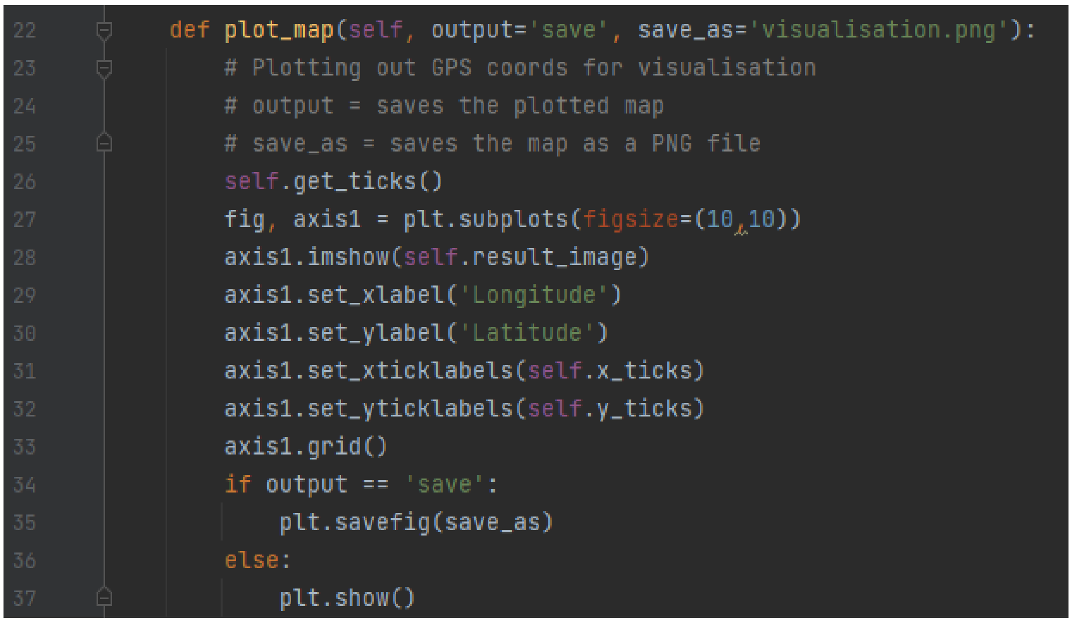
```python
    def __init__(self, data_path, map_path, points):
        # data_path = file containing containing GPS records
        # map_path = location of the map image
        # points = Upper-Left and lower-right GPS points of the map (lat1, lon1, lat2, lon2).
        self.data_path = data_path
        self.points = points
        self.map_path = map_path

        self.result_image = Image
        self.x_ticks = []
        self.y_ticks = []
```

Figure 11 __init__ function code

This function initialises the objects that are created within the class, for this function it would be so that the application is able to locate the file containing the data set, the location of where the static image of the map is located and finally the latitude and longitude of the upper-left and the lower-right points of the map. The latitude and longitude are vital to the application working as it allows the GPS points in the data set are able to be plotted accurately to the in-world location of the user and accurate to the GPS Dataset.

Plot_map

Plot map serves as the way the program can output the saved map once all the coordinates are plotted, it will plot the map in the project folder as ‘visualisation.png’ so that the user can view it.

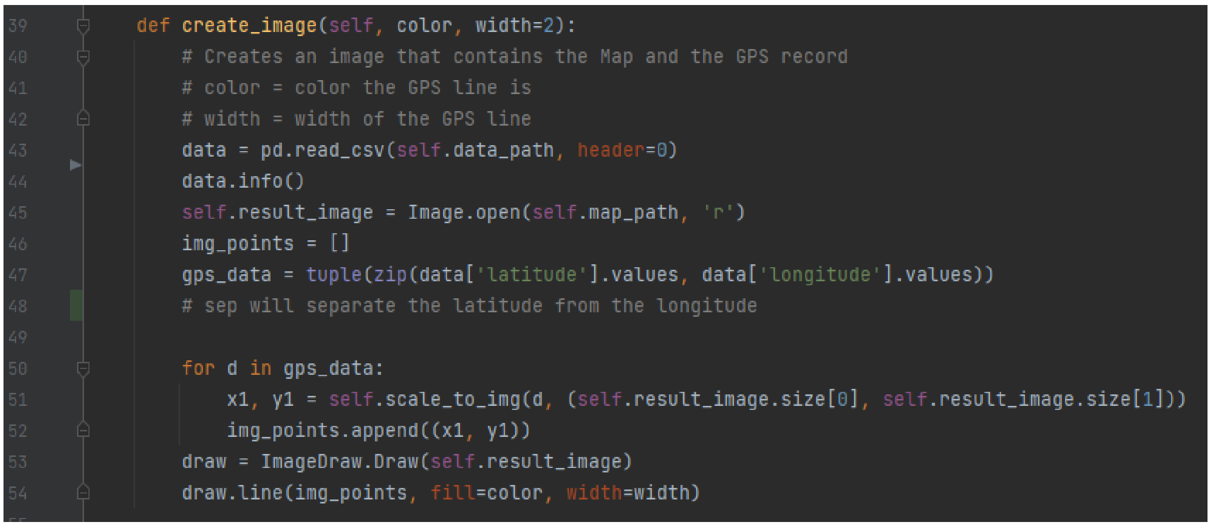


```python
    def plot_map(self, output='save', save_as='visualisation.png'):
        # Plotting out GPS coords for visualisation
        # output = saves the plotted map
        # save_as = saves the map as a PNG file
        self.get_ticks()
        fig, axis1 = plt.subplots(figsize=(10,10))
        axis1.imshow(self.result_image)
        axis1.set_xlabel('Longitude')
        axis1.set_ylabel('Latitude')
        axis1.set_xticklabels(self.x_ticks)
        axis1.set_yticklabels(self.y_ticks)
        axis1.grid()
        if output == 'save':
            plt.savefig(save_as)
        else:
            plt.show()
```

Figure 12 plot_map function code

Create_image

Create image has the task of reading the .csv file and extracting the gps data so that it can be stored into the program and then made into a line which takes information such as colour and width from the GPS_Visualisation function.

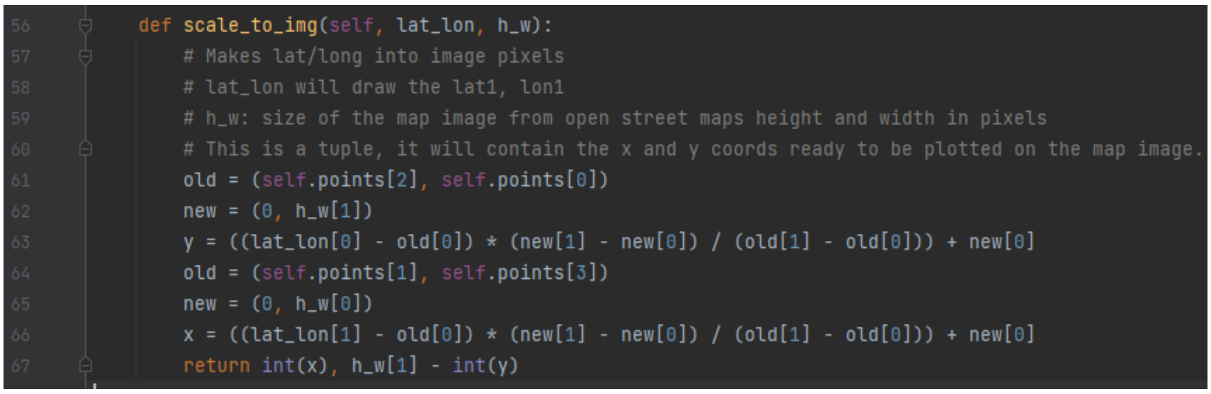


```python
def create_image(self, color, width=2):
    # Creates an image that contains the Map and the GPS record
    # color = color the GPS line is
    # width = width of the GPS line
    data = pd.read_csv(self.data_path, header=0)
    data.info()
    self.result_image = Image.open(self.map_path, 'r')
    img_points = []
    gps_data = tuple(zip(data['latitude'].values, data['longitude'].values))
    # sep will separate the latitude from the longitude

    for d in gps_data:
        x1, y1 = self.scale_to_img(d, (self.result_image.size[0], self.result_image.size[1]))
        img_points.append((x1, y1))
    draw = ImageDraw.Draw(self.result_image)
    draw.line(img_points, fill=color, width=width)
```

Figure 13 create_image function code

Scale_to_img

Scale to image allows the program to acknowledge the image file which has been input with the coordinates asked so that it can plot the data out accurately. It will scale the image from the coordinates input into pixels, these pixels become equal to the in-world coordinates so that the points get plotted correctly.

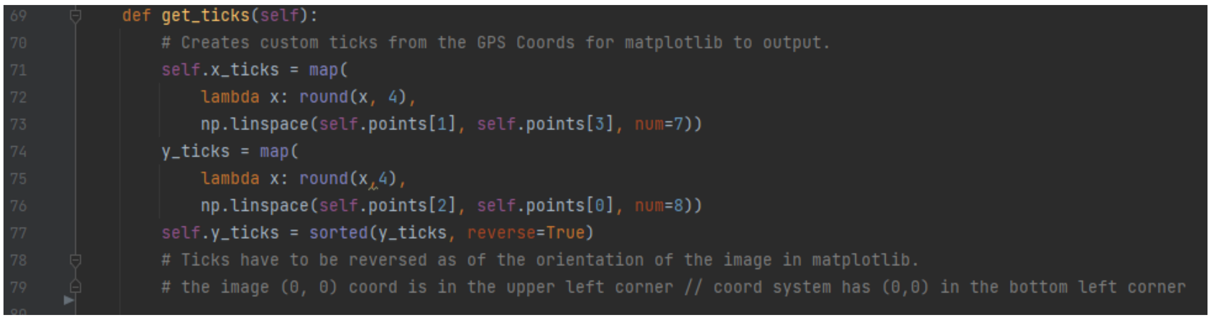


```python
def scale_to_img(self, lat_lon, h_w):
    # Makes lat/long into image pixels
    # lat_lon will draw the lat1, lon1
    # h_w: size of the map image from open street maps height and width in pixels
    # This is a tuple, it will contain the x and y coords ready to be plotted on the map image.
    old = (self.points[2], self.points[0])
    new = (0, h_w[1])
    y = ((lat_lon[0] - old[0]) * (new[1] - new[0]) / (old[1] - old[0])) + new[0]
    old = (self.points[1], self.points[3])
    new = (0, h_w[0])
    x = ((lat_lon[1] - old[0]) * (new[1] - new[0]) / (old[1] - old[0])) + new[0]
    return int(x), h_w[1] - int(y)
```

Figure 14 scale_to_img function code

Get_ticks

This enables the matplotlib to plot the points on the map using its features, the only issue is that the (0,0) coordinates of the image we are inputting to the application are in the top left corner, while under matplotlib it is in the bottom left corner. To combat this, the application needs to flip the image so the (0,0) positions match. Note, the (0,0) coordinate does change value based off of what the user input via the GUI but the application needs to match that top left coordinate, so it has to be forced to flip.



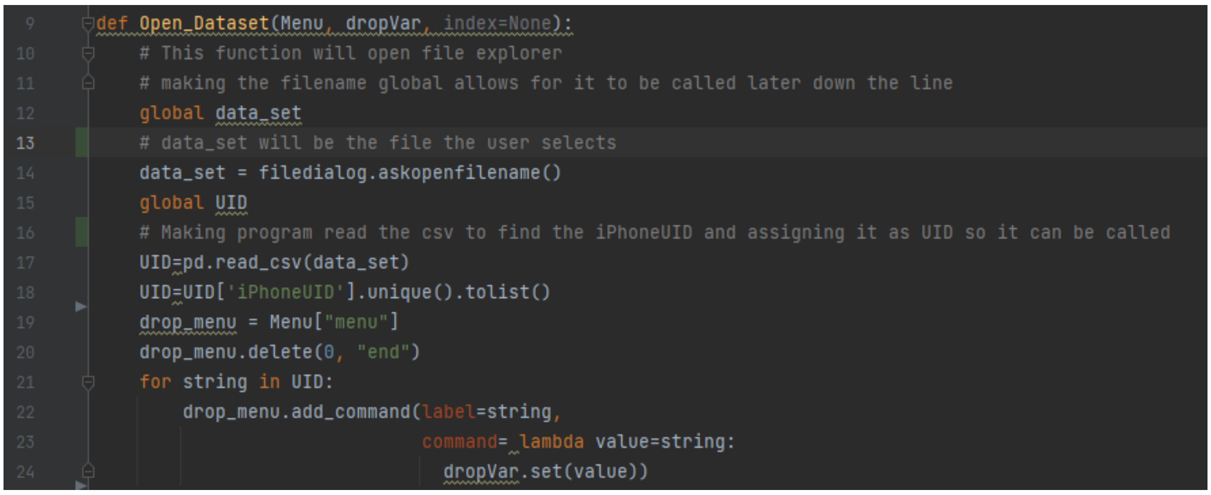
```python
    def get_ticks(self):
        # Creates custom ticks from the GPS Coords for matplotlib to output.
        self.x_ticks = map(
            lambda x: round(x, 4),
            np.linspace(self.points[1], self.points[3], num=7))
        y_ticks = map(
            lambda x: round(x,4),
            np.linspace(self.points[2], self.points[0], num=8))
        self.y_ticks = sorted(y_ticks, reverse=True)
        # Ticks have to be reversed as of the orientation of the image in matplotlib.
        # the image (0, 0) coord is in the upper left corner // coord system has (0,0) in the bottom left corner
```

Figure 15 get_ticks function code

GUI.py

The GUI was a key development on the original code, it allowed for the user to interact with the program by enabling them to input data into an external point for use within the program's code. The original code required the user to input the latitude and longitudes of the map, the data path for the dataset and the map, this would have to be edited each time the user wanted to change any of the inputs and that would have to be done via an IDE which would be time consuming.

Open_Dataset

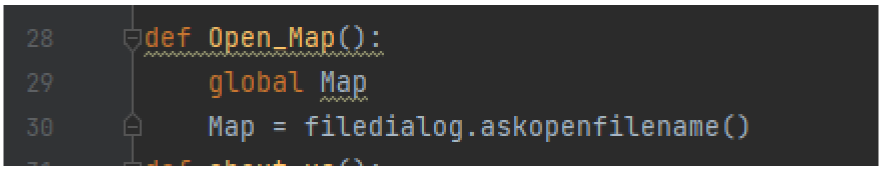


```
def Open_Dataset(Menu, dropVar, index=None):
    # This function will open file explorer
    # making the filename global allows for it to be called later down the line
    global data_set
    # data_set will be the file the user selects
    data_set = filedialog.askopenfilename()
    global UID
    # Making program read the csv to find the iPhoneUID and assigning it as UID so it can be called
    UID=pd.read_csv(data_set)
    UID=UID['iPhoneUID'].unique().tolist()
    drop_menu = Menu["menu"]
    drop_menu.delete(0, "end")
    for string in UID:
        drop_menu.add_command(label=string,
                              command= lambda value=string:
                                dropVar.set(value))
```

Figure 16 Open Dataset Code

This function is responsible for opening the selected dataset by the user so that it can be read and have data extracted. In this case it is reading the dataset for the 'iPhoneUID' which is key for the program to split up the dataset so that it can plot just the GPS Coordinates of a singular phone ID. The program then updates a dropdown list of the IDs so the user can select them, this list will remain blank until the dataset is selected and then it will show the ID of the iPhone.

Open_Map

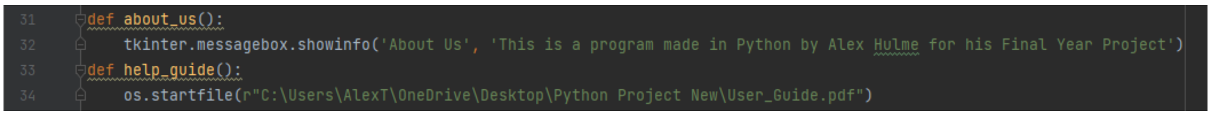


```
def Open_Map():
    global Map
    Map = filedialog.askopenfilename()
```

Figure 17 Open Map Button Code

As previously explained, this part of the GUI is enabling the user to be able to select the map file. It is called once the button on the GUI is pressed, it opens the file explorer so the user can select a file, this then assigns a value to the variable 'Map' which as a global variable can be called outside of this function.

About_us & help_guide

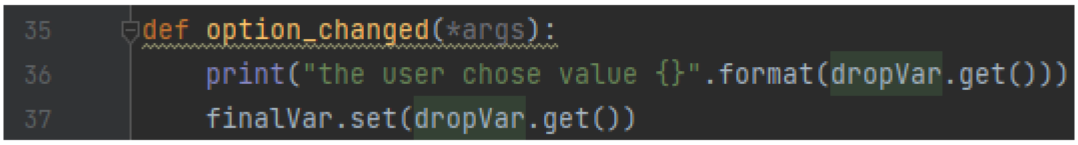


```python
def about_us():
    tkinter.messagebox.showinfo('About Us', 'This is a program made in Python by Alex Hulme for his Final Year Project')
def help_guide():
    os.startfile(r"C:\Users\AlexT\OneDrive\Desktop\Python Project New\User_Guide.pdf")
```

Figure 18 about_us and help_guide code

To make the program more user friendly, an about us section was implemented which would allow the user to see who made the program and what language it was in. This is commonly saw this used in applications and so was added as a minor feature. The help_guide links the user to be able to see how to use the program, it uses 'os' to force the operating system to open the pdf file in a readable view.

Option_changed

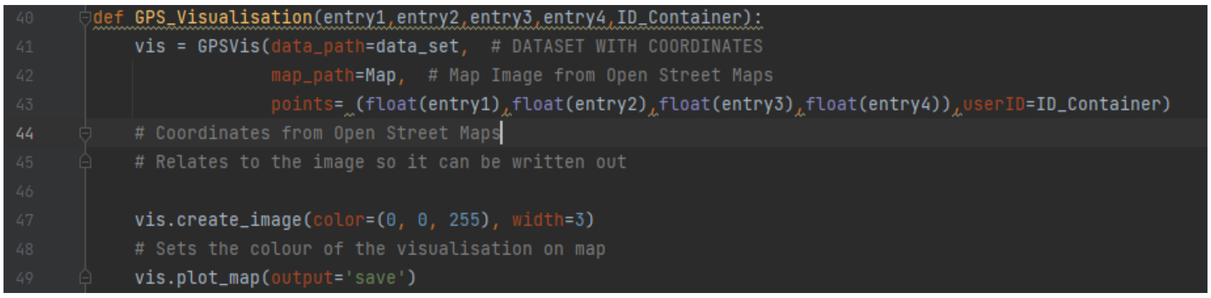


```python
def option_changed(*args):
    print("the user chose value {}".format(dropVar.get()))
    finalVar.set(dropVar.get())
```

Figure 19 option_changed code

This function ensures the program is passed the right iPhone ID based on what the user has chosen from the dropdown list is stored. This will append the value to the dropVar variable, the finalVar will then be set as the value assigned to dropVar for later use.

GPS_Visualisation



```python
def GPS_Visualisation(entry1,entry2,entry3,entry4,ID_Container):
    vis = GPSVis(data_path=data_set,  # DATASET WITH COORDINATES
                 map_path=Map,  # Map Image from Open Street Maps
                 points= (float(entry1),float(entry2),float(entry3),float(entry4)),userID=ID_Container)
    # Coordinates from Open Street Maps
    # Relates to the image so it can be written out

    vis.create_image(color=(0, 0, 255), width=3)
    # Sets the colour of the visualisation on map
    vis.plot_map(output='save')
```

Figure 20 GPS_Visualisation Code

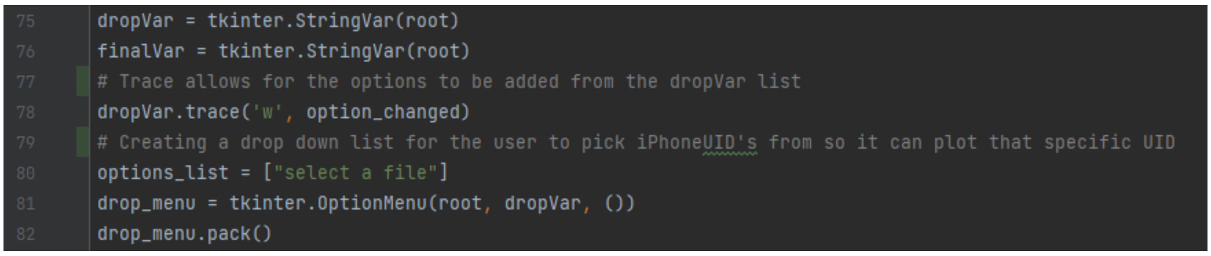
GPS Visualisation is the main bulk of the visualisation for the program, it is a bridging function between the GUI with the user input and the main program that will use the inputs to calculate and plot on the map.

It works by making a variable called 'vis' and this contains the data path for the dataset and map as well as the points which the user will input via the GUI. It holds these numbers as a float, this is because coordinates work by having a degree value followed by minutes so will always hold a value that has a decimal place. It would be unable to use 'int' because of an integer requiring it to be a whole number so it would then lose accuracy. These points need to be accurate as they mark the boundaries (top left, lower right) of the map, so the program can then plot the points to a higher degree of accuracy.

The points also include the userID, this is so that the program can find the data which relates to the userID that has been selected by the user, otherwise the program would just plot every point in the data set regardless of the ID.

‘vis.create_image’ is where the plotted line would be, the colour of the plotting line can be changed within the code as it is using the RGB scale. With the current values in the line would appear as a solid Blue but this is just one of over 16 million colours which can be created in the RGB scale. I have also been able to control the width of the line that will be plotted to 3 pixels, if it needs to be more prominent, it can have the value increased but this would lose detail if the area of operation is small or the base image has a lot of streets nearby and is scaled to cover a larger area.

dropVar



```
dropVar = tkinter.StringVar(root)
finalVar = tkinter.StringVar(root)
# Trace allows for the options to be added from the dropVar list
dropVar.trace('w', option_changed)
# Creating a drop down list for the user to pick iPhoneUID's from so it can plot that specific UID
options_list = ["select a file"]
drop_menu = tkinter.OptionMenu(root, dropVar, ())
drop_menu.pack()
```

Figure 21 dropVar code

dropVar is initially assigned as a string variable which contains an empty string, it obtains a value when the user uploads the dataset as it will hold the contents of the dropdown list. As seen earlier, once the user selects what user ID they want from the dropdown list the program assigns this value to finalVar which is also an empty string.

The trace function it used to monitor a specific event which is related to the GUI, so it will monitor the options changed function.

Testing Regime

Overview

As outlined in the indicative test plan, use of the waterfall methodology to conduct testing and development was planned. However due to the nature of the project where there will be constant development and testing, it was decided that it would be more efficient to use the Agile methodology as it would allow for more testing on the go when a new feature is implemented into the project to ensure it cooperates with the already existing parts of the project that had been implemented earlier. This would save me time by allowing me to troubleshoot sections quickly by doing it stage by stage.

Unit Testing

The unit testing will ensure that the code being developed and modified from other sources meets quality standards before it is implemented and later deployed. This will ensure that the development is streamlined, and any errors are identified within this testing so that they can be rectified to enable the project to work as planned.

Black Box Testing

Black box testing enables the testing of the functional parts of the project, the testing method is there to identify the correctness of the behaviours within the applications from what the user would see. This allows us to test against our expected outcomes and actual outcomes, if a test does not meet the standards that are expected of it, then it allows opportunity to go back and see why this outcome happened if it negatively impacted the project or if it positively affected the project. The approach is also well suited to exercising the applications in a user case scenario, in which individuals can be asked to test it on behalf of the development team and compare their results to each other to see if the project is behaving the same across the board, while removing any discrepancies.

Please see the testing table that is in my Appendices 1.1.

Critical Evaluation

Project Evaluation

The aim of the project was to show someone how their activities could be tracked and information about them could be compromised through these actions. For example, if they visit a place of worship, what religion are they? Do they shop at a specific shop, say a halal or kosher butchers or do they go a fish market on a Friday? Do they visit a care home where their elderly family member might be or do they frequent a park? What information could be found by them that could be a breach of their privacy and their personal security?

As with the original aim, it was to improve individual's security. On reflection of creating the software, it was a tool that was created to assist people who felt their personal security might be at risk, but it could easily be used for malicious purposes. If someone was to put a tracker on someone's phone through a malicious software being downloaded and inputted the data into the software it would be easy for them to map out the whereabouts of the individual and analyse the data for common patterns to exploit.

From conducting an analysis on the iPhone 'a841f74e620f74ec443b7a25d7569545' that it was a large amount of ground covered, this made the scaling of the map lose some of its detail to allow for it to plot. This means any analysis for key points would be difficult from the map alone, it would require the user to have the original map open to do a side-by-side comparison. As well as the resolution of the map, there is the issue that with large datasets. The points constantly cross and individual detail is lost when multiple high traffic areas are plotted. If this data set was smaller it would plot with detail but the entire data set covers a time period from 09/2012 through to 04/2013, this is a large period of time to plot and would not give any actionable data for day to day but would give off the areas of high traffic in which the user could be found to be operating in.

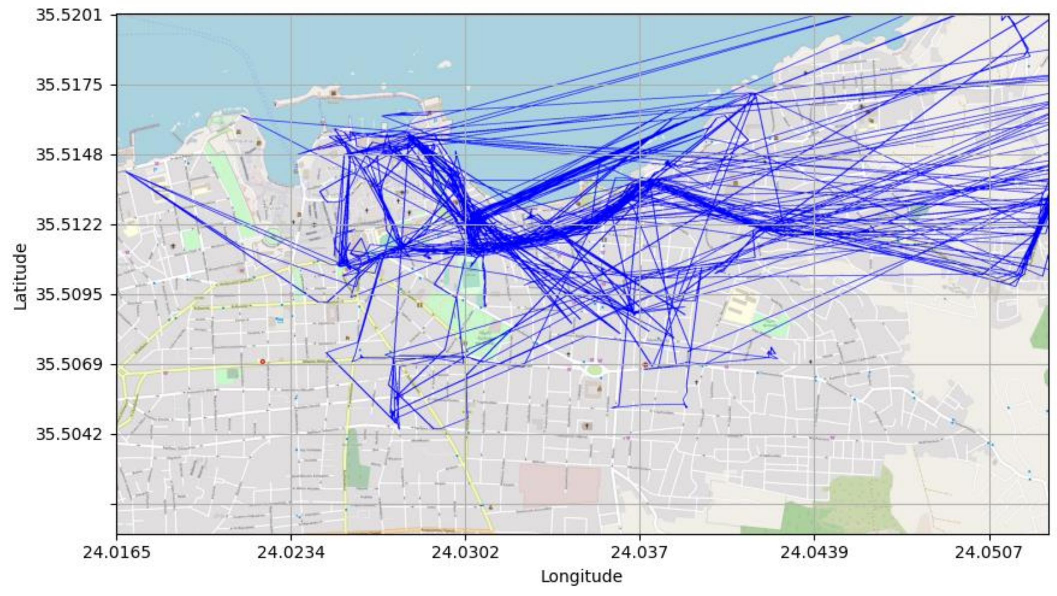


Figure 22 Plotting of Data for iPhone a841f74e620f74ec443b7a25d7569545

By doing a process of elimination and looking at where the user has the heaviest traffic, I was able to crop down the area I was operating in to look at the data.

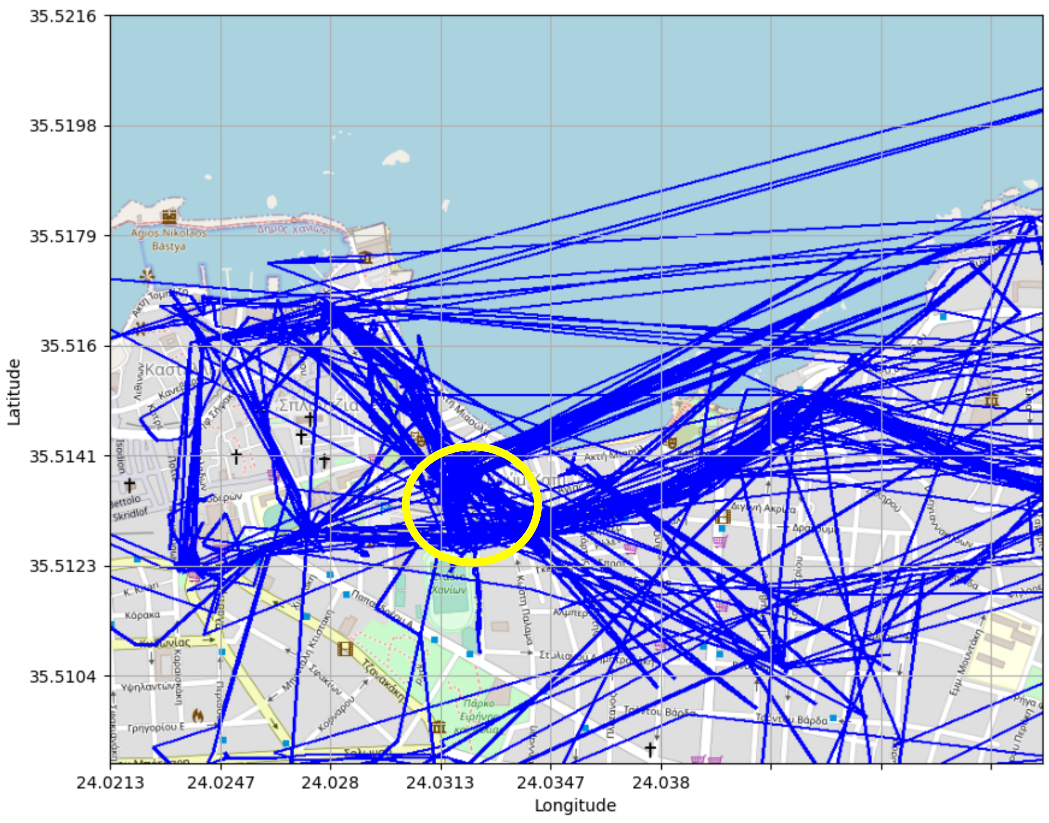


Figure 23 Heavy Traffic of a841f74e620f74ec443b7a25d7569545

From analysing the data of user 'a841f74e620f74ec443b7a25d7569545' I was able to find out they were in the city of Chania, Crete (a Greek Island off the mainland in the Mediterranean.) The user frequented around the Old Town and Splantzia area of the city and along the El. Venizelou street towards the west of the Athletics Centre seen just below the yellow circle in Figure 23. The there was a lot of crossing of GPS points and it naturally created a triangulation of positions around Nikiforou Foka, Kiprou and El Venizelou streets which could suggest the user was staying within the Kriti Hotel that is within this area or in a property that is in the same area as seen in Figure 24.

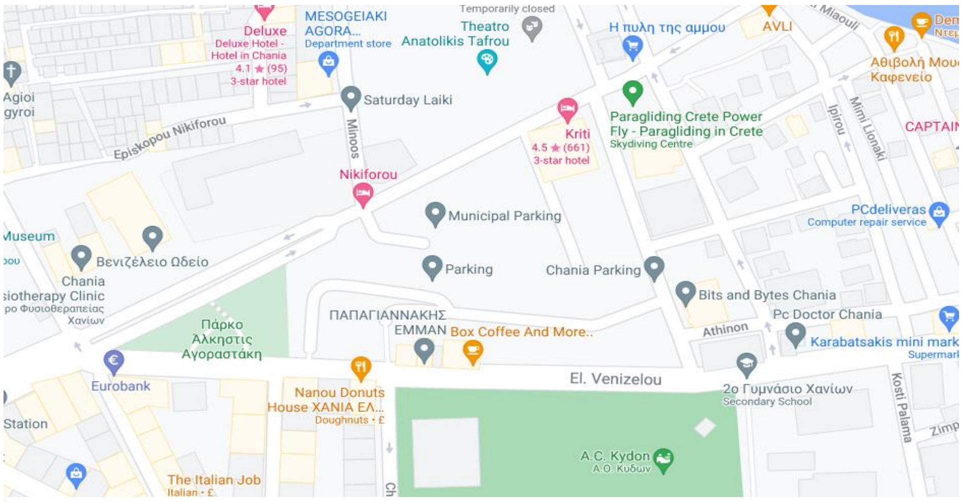


Figure 24 Google Maps Capture of Area of Operations

Another application of this project could be that it is used by armed forces and law enforcement to track and monitor the area of operation of HVT (High Value Targets) or informants (Katina Michael, 2006). From this information you could see roads and towns which are highly travelled to and base that off of other intelligence to find other HVTs that could be acting in relation to this individual. (Abubakar Sani, 2015) For example, if an enemy combatant was using a certain application and the GPS data was extracted and put into the software created it could show that they have visited a town that is already under suspicion for housing a leader of a resistance movement or a known warlord and allow for actionable intelligence to be disseminated to other units operating in the area. It could even show the locations of stashes of weapons or drugs if they are in random areas in which the user has only frequented once or twice, which would show on the map for having a low are of traffic from that individual based on their GPS data.

Not all the aims of the project were achieved, however the current version of the project created a base for further development by myself and other individuals who are interested in this topic. As the project can now plot GPS coordinates, the further development would be to enable this on a non-static map and allow for the user to just upload a dataset and it make a working map for the user. Another point for future development is being able to split the times you wish data to be shown, from my testing I realised some datasets were 6 months of GPS data and would plot out the entire set for that individual ID. It should be an easy fix to allow the user to pick a timeframe in future versions of this.

Finally, when it comes to further applications a useful feature could be the allowance of comparison of data between different IDs. This would be useful in the HVT scenario previously explained, this could be combined with the time split to provide information if two or more individuals met in person or were at the same location at the same time.

System Features

The system features a GUI, something which the original code did not have. This GUI was simplistic and made it less complicated for the user to navigate, it did not have multiple pages and only had what was necessary for it to function. It did not need any colour scheme as this could make it difficult for people to use the program properly, it had buttons that allowed the user to access the data they needed which made it easier than them having to change it each time in the IDE.

This process would be confusing for the user for the first time so included a user guide which was accessible from a help tab at the top, which is a thing of familiarity as this is in many programs that are already out. The user guide would explain where to get the map from and the coordinates that are needed and what they are. Unfortunately, this part could not be automated as each map would have different coordinates and it would be down to the user to successfully input the data for execution, this is the weak point of the program.

Development Evaluation

Development Approach

I believe my approach was very rushed, my time management was not the best that it could have been. I was focusing on finishing my other module work to open up more time to focus on my final year project, out of my second term modules I managed to complete two of the three in a short period of time before getting bogged down by only focusing on one module at a time. In hindsight I should have split my time a bit more.

My initial approach was to break it down into workable sections, such as setting up the PostGres and PostGIS database so that it was working, I instead neglected this and left it later than I should have and it more than likely resulted in the database not being constructed correctly and causing crashes or errors.

My development was fitting around sustainability of my own well-being, I utilised downtime of university when laboratories were shut and when the University Officers Training Corps went into exam hiatus to focus on the main bulk of my project. This allowed for me to achieve more without impacting my job or final module waiting to be completed.

Academic Advancement

Throughout this project I feel like I have learnt how to utilise my time to the fullest and how to balance a project even in unusual times such as the COVID-19 pandemic and how to communicate with a supervisor on the process of this project. Even with limited access to the normal resources I would have and being able to go in person and ask questions, it was challenging to sometimes get the point across but that I learnt was due to what is written in text would sound completely different to how you would say it in person.

I have also learnt to be flexible, initially I was creating a project that used live data to track someone as they went and I saw this was too difficult, then went out and discussed the idea to find alternatives. When one alternative did not work, I would try minimising any of the losses I have had by seeing what resources or skills I could transfer over. For example, when I realised my PostGres database did not communicate with my software, I scrapped it and saw how I could still use Open Street Maps to my advantage as it was a valuable tool.

From a more personal development standpoint, I have learnt how to manage stress under a stressful time period in life (COVID-19 pandemic) and allowed myself to take time to think on the problem rather than expecting myself to instantly know the solution to a problem. This sometimes meant taking time away for a day or two and attacking the problem on another day.

I was also able to practice developing my skills in presenting through my job, which I then used some of my work colleagues to talk about my presentation as a group and gain more confidence in preparation for my viva presentation.

Tool Evaluation

Development Support

For the development of my project I began using the base Python IDE, after speaking with other students when I was stuck and showed them my code; I was introduced to PyCharm. PyCharm being an IDE which made it easier to implement code, one of the main features I used was the predictive text; when I declared a variable or a function earlier in the code I could begin typing and it would give me multiple options for the word I was typing which saved time. It also had an active bug checker, so as I coded it would look for syntax errors or unknown commands and highlighted them with detail which the base IDE would fail to do.

Finally, PyCharm allowed direct commits and pushes to my GitHub account so that I could add commit notes and not have to leave a trail of update comments at the bottom of the code. It also enabled others to see my work and give a fresh set of eyes to spot any errors within my code, while also being able to provide me with some fixes to ensure the project would function as planned.

References

Abubakar Sani, Z. S. P. H. A. U., 2015. Location Information: Another Perspective of Intelligence Gathering for Minimizing Terrorism in Nigeria. *International Journal of Scientific & Engineering Research,* 6(9), p. 244.

Akinode, J., 2011. Improving National Security Using GPS Tracking System Technology. *Mediterranean Journal of Social Sciences,* 2(5).

Baijal, R., 2009. *GPS: A Military Perspective,* s.l.: s.n.

blue_beetle, 2010. *User Driven Discontent,* s.l.: MetaFilter.

Clifton, K., 2020. *Met Police Chief condemns assaults on officers during London Black Lives Matter protests.* London: Evening Standard.

Colour Blind Awareness, n.d. *Colour Blindness.* [Online]
Available at: https://www.colourblindawareness.org/colour-blindness/
[Accessed 25 02 2021].

ENCA, 2019. *SAMA: Rising attacks on healthcare workers are a concern.* [Online]
Available at: https://www.enca.com/analysis/rising-attacks-healthcare-workers-concern
[Accessed 16 03 2021].

Ghose, D., 2017. *Determination of Transportation Modes Using Mobile Phones,* Karnataka: Manipal University.

Google Cloud Platform, 2014. *openmobiledata_public,* s.l.: s.n.

Home Office, 2020. *Proscribed Terrorist Organisations.* [Online]
Available at:
https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/901434/20200717_Proscription.pdf
[Accessed 16 03 2021].

IOL, 2020. *Attacks on paramedics have more than tripled.* [Online]
Available at: https://www.iol.co.za/weekend-argus/news/attacks-on-paramedics-have-more-than-tripled-2f274464-9949-4599-b80b-5aa3a546aabb
[Accessed 16 03 2021].

Katina Michael, A. M. M. M., 2006. *The Emerging Ethics of Humancentric GPS Tracking and Monitoring,* s.l.: University of Wollongong.

Katina Michael, R. C., 2013. Location and Tracking of Mobile Devices: Uberveillance stalks the streets. *Computer Law & Security Review,* 29(3), pp. 216-228.

MI5, 2021. *Threat Levels.* [Online]
Available at: https://www.mi5.gov.uk/threat-levels
[Accessed 16 03 2021].

Microsoft Research Asia, 2011. *Geolife GPS trajectory dataset - User Guide,* Beijing: Microsoft.

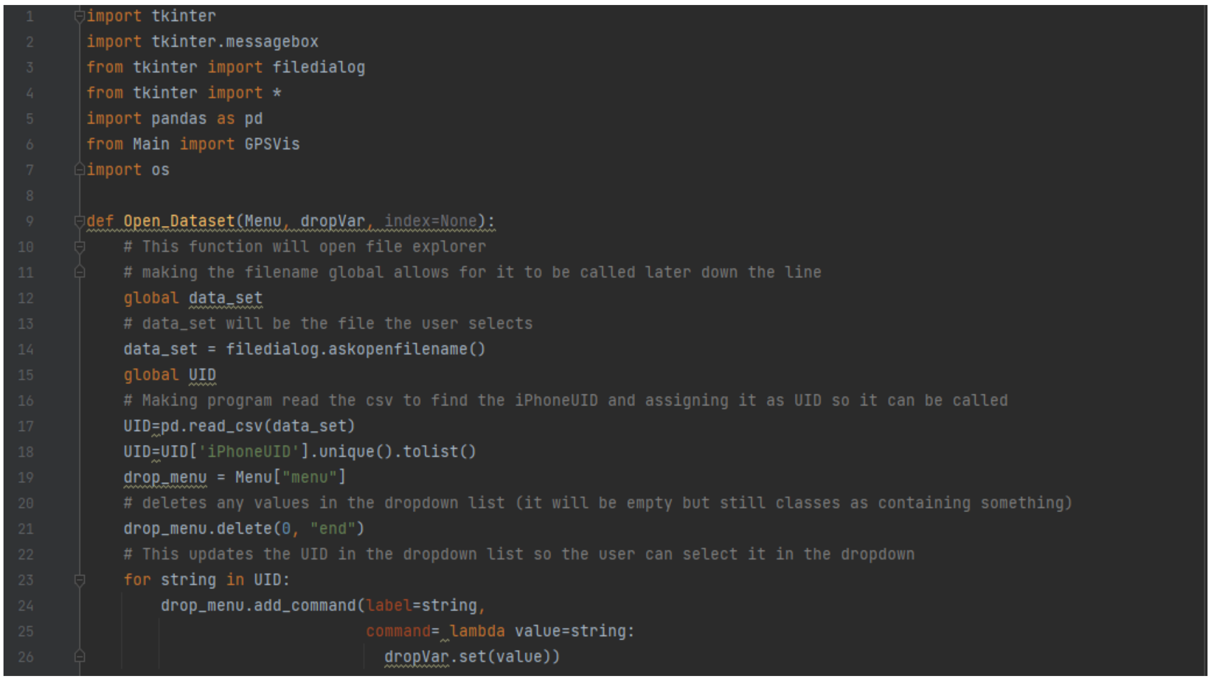
The Guardian, 2016. *Jo Cox killed in 'brutal, cowardly' and politically motivated murder, trial hears.* [Online]
Available at: https://www.theguardian.com/uk-news/2016/nov/14/jo-cox-killed-in-politically-

motivated-murder-trial-thomas-mair-hears
[Accessed 16 03 2021].

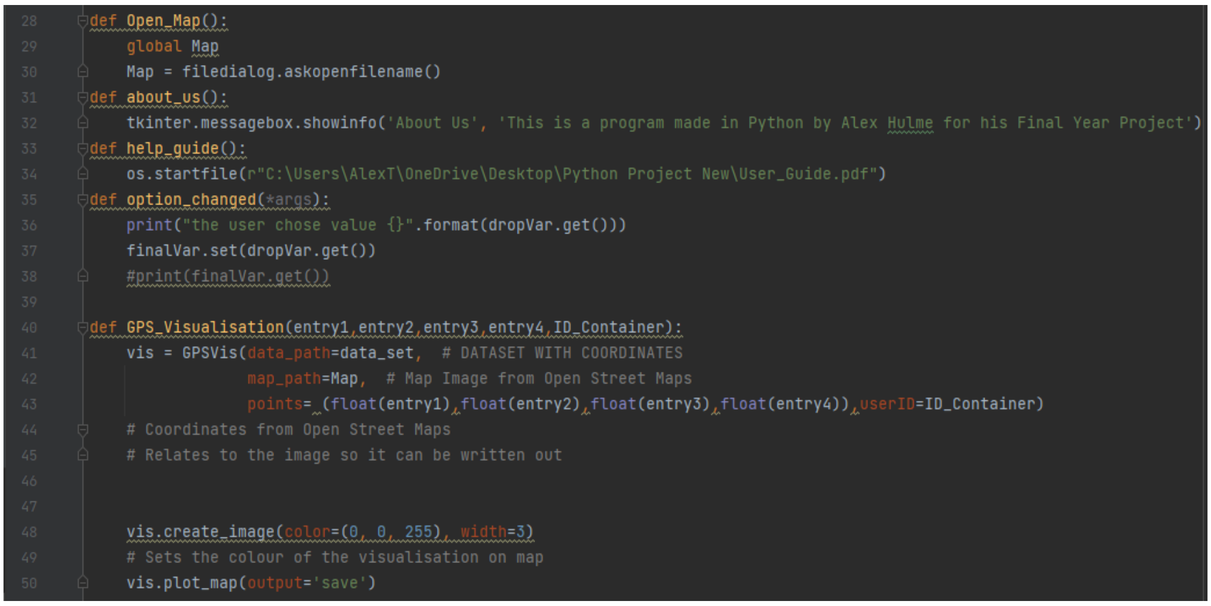
Tisljaric, L., n.d. *Simple GPS data visualization using Python and Open Street Maps,* s.l.: Towards data science.

Code Samples

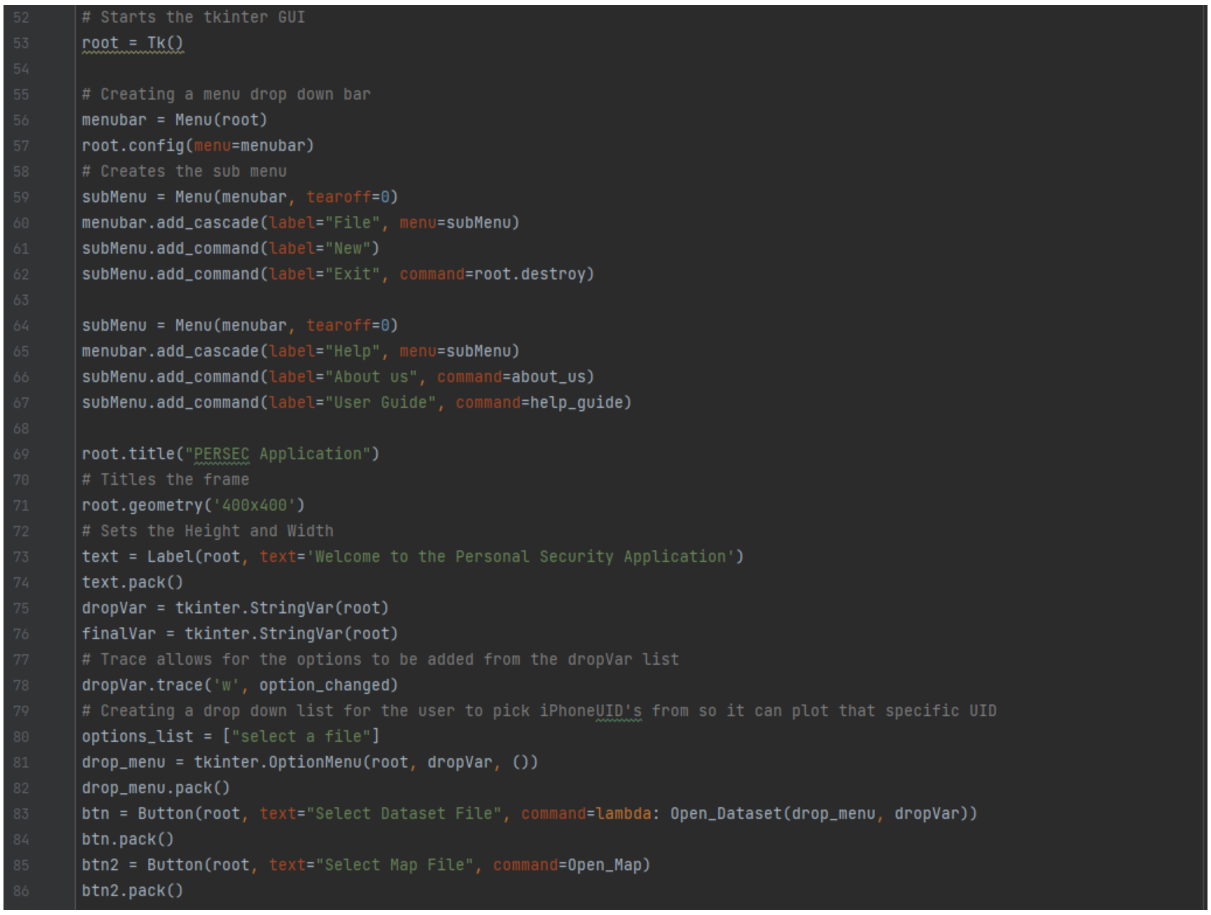
GUI.py



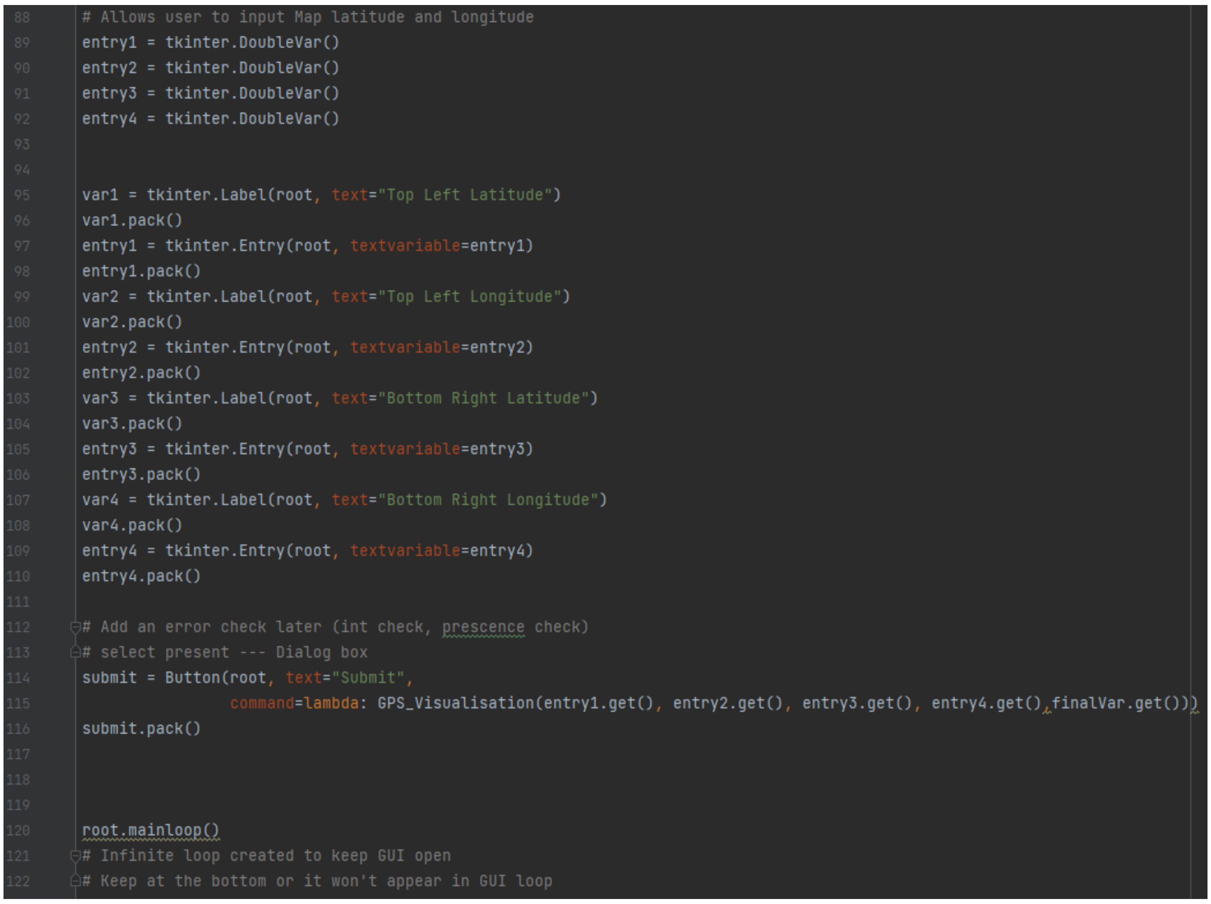
```python
import tkinter
import tkinter.messagebox
from tkinter import filedialog
from tkinter import *
import pandas as pd
from Main import GPSVis
import os

def Open_Dataset(Menu, dropVar, index=None):
    # This function will open file explorer
    # making the filename global allows for it to be called later down the line
    global data_set
    # data_set will be the file the user selects
    data_set = filedialog.askopenfilename()
    global UID
    # Making program read the csv to find the iPhoneUID and assigning it as UID so it can be called
    UID=pd.read_csv(data_set)
    UID=UID['iPhoneUID'].unique().tolist()
    drop_menu = Menu["menu"]
    # deletes any values in the dropdown list (it will be empty but still classes as containing something)
    drop_menu.delete(0, "end")
    # This updates the UID in the dropdown list so the user can select it in the dropdown
    for string in UID:
        drop_menu.add_command(label=string,
                              command= lambda value=string:
                                dropVar.set(value))
```



```python
def Open_Map():
    global Map
    Map = filedialog.askopenfilename()
def about_us():
    tkinter.messagebox.showinfo('About Us', 'This is a program made in Python by Alex Hulme for his Final Year Project')
def help_guide():
    os.startfile(r"C:\Users\AlexT\OneDrive\Desktop\Python Project New\User_Guide.pdf")
def option_changed(*args):
    print("the user chose value {}".format(dropVar.get()))
    finalVar.set(dropVar.get())
    #print(finalVar.get())

def GPS_Visualisation(entry1,entry2,entry3,entry4,ID_Container):
    vis = GPSVis(data_path=data_set,  # DATASET WITH COORDINATES
                 map_path=Map,  # Map Image from Open Street Maps
                 points= (float(entry1),float(entry2),float(entry3),float(entry4)),userID=ID_Container)
    # Coordinates from Open Street Maps
    # Relates to the image so it can be written out


    vis.create_image(color=(0, 0, 255), width=3)
    # Sets the colour of the visualisation on map
    vis.plot_map(output='save')
```

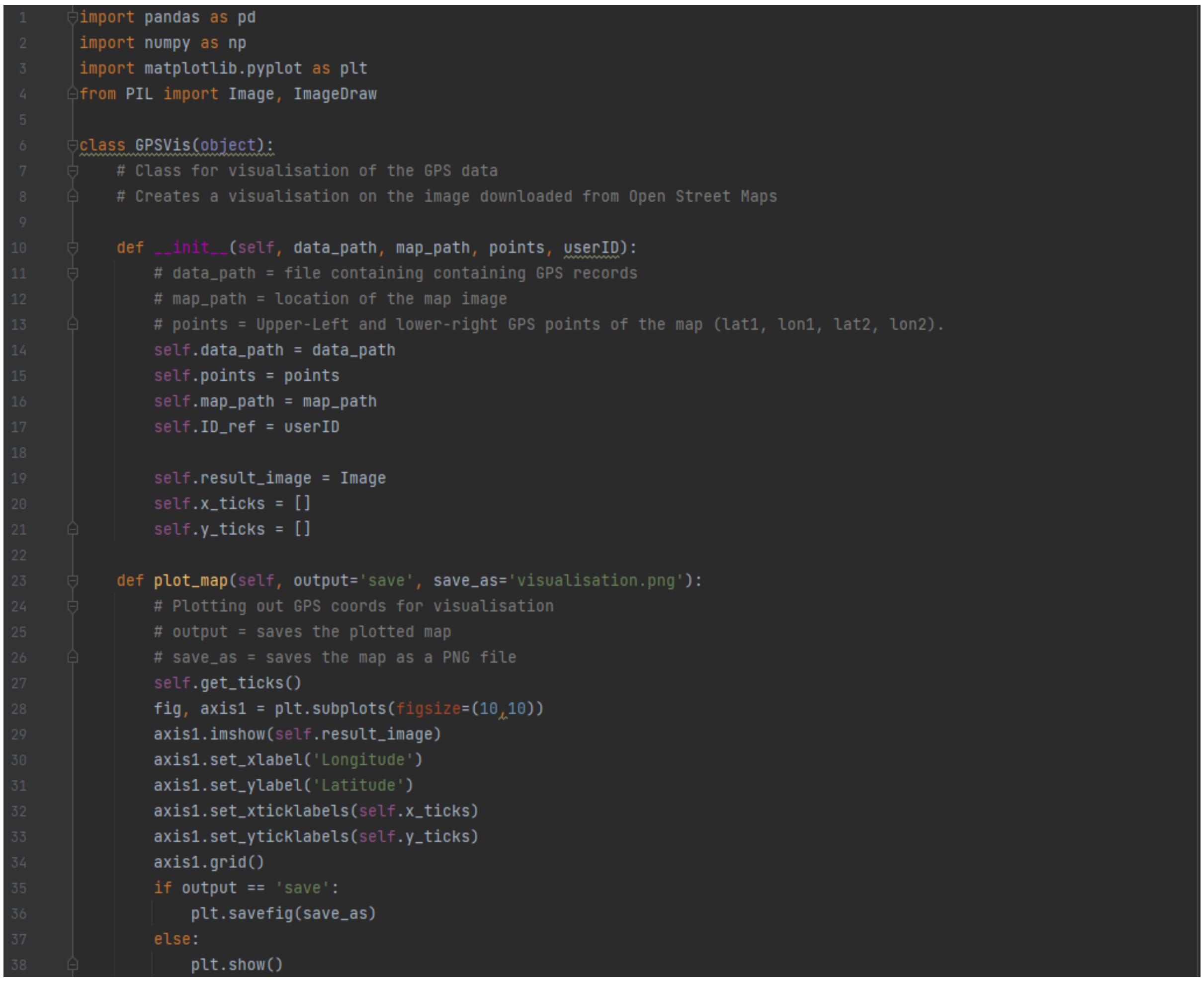


```python
# Starts the tkinter GUI
root = Tk()

# Creating a menu drop down bar
menubar = Menu(root)
root.config(menu=menubar)
# Creates the sub menu
subMenu = Menu(menubar, tearoff=0)
menubar.add_cascade(label="File", menu=subMenu)
subMenu.add_command(label="New")
subMenu.add_command(label="Exit", command=root.destroy)

subMenu = Menu(menubar, tearoff=0)
menubar.add_cascade(label="Help", menu=subMenu)
subMenu.add_command(label="About us", command=about_us)
subMenu.add_command(label="User Guide", command=help_guide)

root.title("PERSEC Application")
# Titles the frame
root.geometry('400x400')
# Sets the Height and Width
text = Label(root, text='Welcome to the Personal Security Application')
text.pack()
dropVar = tkinter.StringVar(root)
finalVar = tkinter.StringVar(root)
# Trace allows for the options to be added from the dropVar list
dropVar.trace('w', option_changed)
# Creating a drop down list for the user to pick iPhoneUID's from so it can plot that specific UID
options_list = ["select a file"]
drop_menu = tkinter.OptionMenu(root, dropVar, ())
drop_menu.pack()
btn = Button(root, text="Select Dataset File", command=lambda: Open_Dataset(drop_menu, dropVar))
btn.pack()
btn2 = Button(root, text="Select Map File", command=Open_Map)
btn2.pack()
```

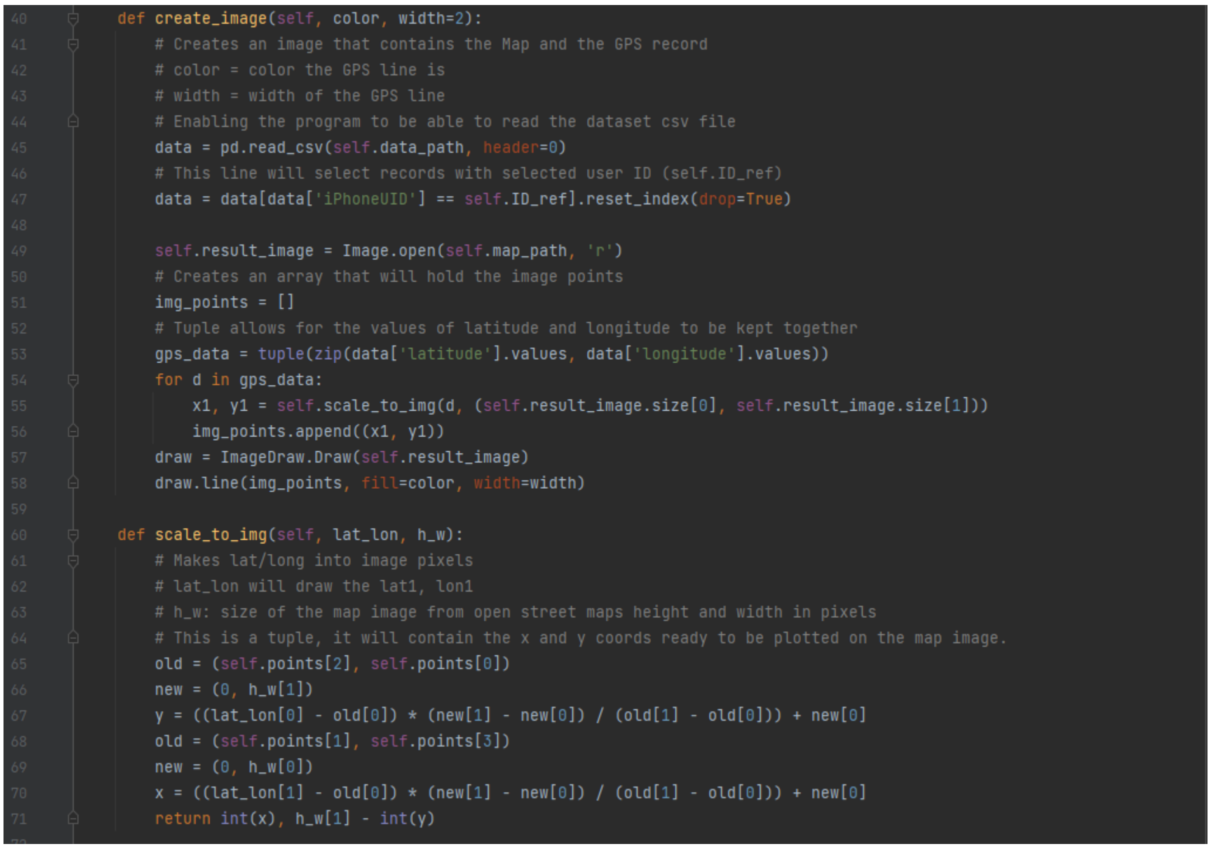


```python
# Allows user to input Map latitude and longitude
entry1 = tkinter.DoubleVar()
entry2 = tkinter.DoubleVar()
entry3 = tkinter.DoubleVar()
entry4 = tkinter.DoubleVar()


var1 = tkinter.Label(root, text="Top Left Latitude")
var1.pack()
entry1 = tkinter.Entry(root, textvariable=entry1)
entry1.pack()
var2 = tkinter.Label(root, text="Top Left Longitude")
var2.pack()
entry2 = tkinter.Entry(root, textvariable=entry2)
entry2.pack()
var3 = tkinter.Label(root, text="Bottom Right Latitude")
var3.pack()
entry3 = tkinter.Entry(root, textvariable=entry3)
entry3.pack()
var4 = tkinter.Label(root, text="Bottom Right Longitude")
var4.pack()
entry4 = tkinter.Entry(root, textvariable=entry4)
entry4.pack()

# Add an error check later (int check, prescence check)
# select present --- Dialog box
submit = Button(root, text="Submit",
                command=lambda: GPS_Visualisation(entry1.get(), entry2.get(), entry3.get(), entry4.get(),finalVar.get()))
submit.pack()



root.mainloop()
# Infinite loop created to keep GUI open
# Keep at the bottom or it won't appear in GUI loop
```

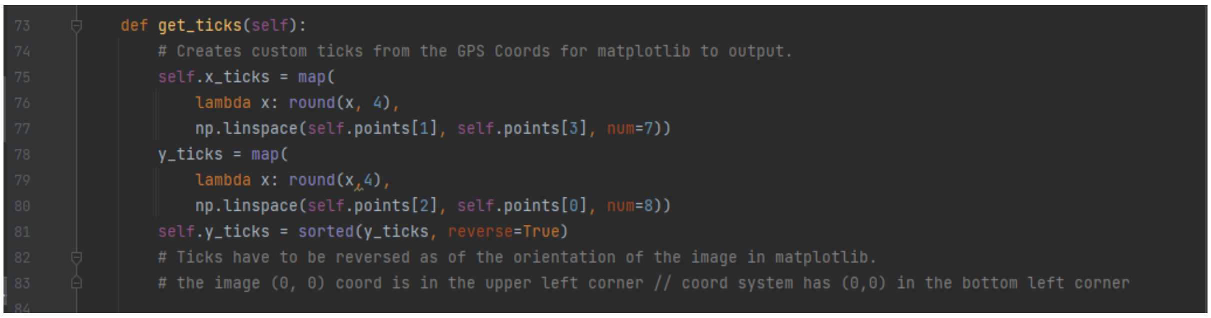
Main.py



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image, ImageDraw

class GPSVis(object):
    # Class for visualisation of the GPS data
    # Creates a visualisation on the image downloaded from Open Street Maps

    def __init__(self, data_path, map_path, points, userID):
        # data_path = file containing containing GPS records
        # map_path = location of the map image
        # points = Upper-Left and lower-right GPS points of the map (lat1, lon1, lat2, lon2).
        self.data_path = data_path
        self.points = points
        self.map_path = map_path
        self.ID_ref = userID

        self.result_image = Image
        self.x_ticks = []
        self.y_ticks = []

    def plot_map(self, output='save', save_as='visualisation.png'):
        # Plotting out GPS coords for visualisation
        # output = saves the plotted map
        # save_as = saves the map as a PNG file
        self.get_ticks()
        fig, axis1 = plt.subplots(figsize=(10,10))
        axis1.imshow(self.result_image)
        axis1.set_xlabel('Longitude')
        axis1.set_ylabel('Latitude')
        axis1.set_xticklabels(self.x_ticks)
        axis1.set_yticklabels(self.y_ticks)
        axis1.grid()
        if output == 'save':
            plt.savefig(save_as)
        else:
            plt.show()
```



```python
    def create_image(self, color, width=2):
        # Creates an image that contains the Map and the GPS record
        # color = color the GPS line is
        # width = width of the GPS line
        # Enabling the program to be able to read the dataset csv file
        data = pd.read_csv(self.data_path, header=0)
        # This line will select records with selected user ID (self.ID_ref)
        data = data[data['iPhoneUID'] == self.ID_ref].reset_index(drop=True)

        self.result_image = Image.open(self.map_path, 'r')
        # Creates an array that will hold the image points
        img_points = []
        # Tuple allows for the values of latitude and longitude to be kept together
        gps_data = tuple(zip(data['latitude'].values, data['longitude'].values))
        for d in gps_data:
            x1, y1 = self.scale_to_img(d, (self.result_image.size[0], self.result_image.size[1]))
            img_points.append((x1, y1))
        draw = ImageDraw.Draw(self.result_image)
        draw.line(img_points, fill=color, width=width)

    def scale_to_img(self, lat_lon, h_w):
        # Makes lat/long into image pixels
        # lat_lon will draw the lat1, lon1
        # h_w: size of the map image from open street maps height and width in pixels
        # This is a tuple, it will contain the x and y coords ready to be plotted on the map image.
        old = (self.points[2], self.points[0])
        new = (0, h_w[1])
        y = ((lat_lon[0] - old[0]) * (new[1] - new[0]) / (old[1] - old[0])) + new[0]
        old = (self.points[1], self.points[3])
        new = (0, h_w[0])
        x = ((lat_lon[1] - old[0]) * (new[1] - new[0]) / (old[1] - old[0])) + new[0]
        return int(x), h_w[1] - int(y)
```



```python
    def get_ticks(self):
        # Creates custom ticks from the GPS Coords for matplotlib to output.
        self.x_ticks = map(
            lambda x: round(x, 4),
            np.linspace(self.points[1], self.points[3], num=7))
        y_ticks = map(
            lambda x: round(x,4),
            np.linspace(self.points[2], self.points[0], num=8))
        self.y_ticks = sorted(y_ticks, reverse=True)
        # Ticks have to be reversed as of the orientation of the image in matplotlib.
        # the image (0, 0) coord is in the upper left corner // coord system has (0,0) in the bottom left corner
```

Appendices

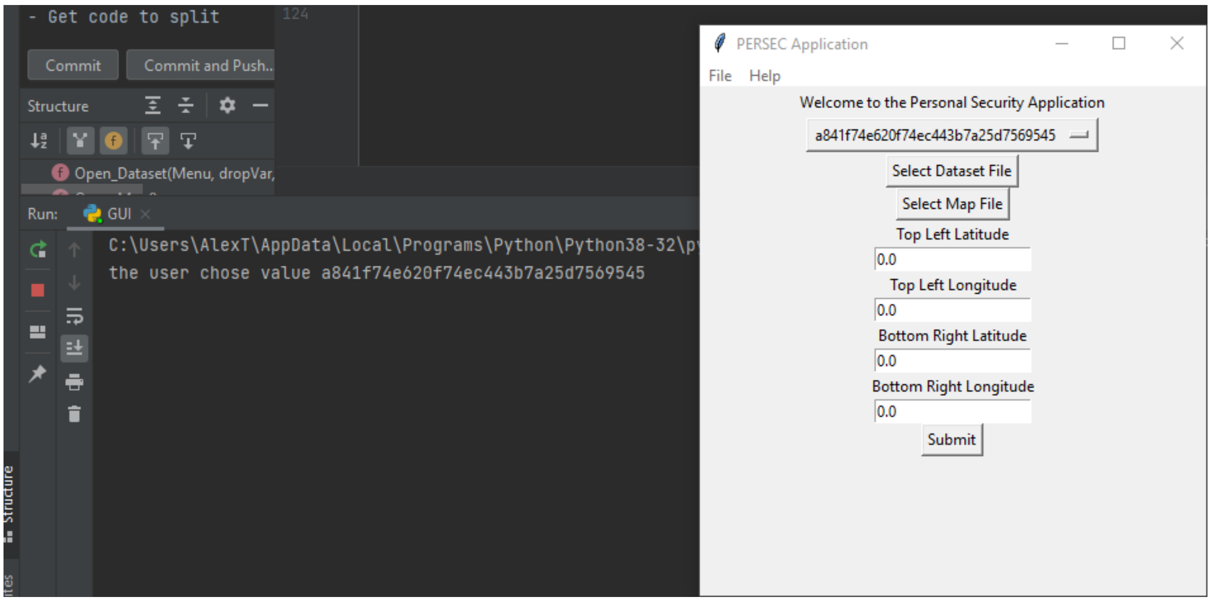
Appendix 1.1 - Testing

Case	Summary	Process	Actual Result	Expect Result	Pass/Fail
1	GUI launches when application is launched.	GUI window will open as the first process and display a 400 x 400 px square.	GUI launches once application is ran.	GUI opens up once application is ran.	🟩
2	‘Select Dataset’ Button opens file explorer when clicked.	Button is clicked by user which executes command code to open the file explorer.	Explorer window opens to find the data set.	Button opens the file explorer.	🟩
3	‘Select Map’ Button opens file explorer when clicked.	Button is clicked by user which executes command code to open the file explorer.	Explorer window opens to find map file location.	Button opens the file explorer.	🟩
4	Dropdown list is able to be interacted with	Drop down list when clicked should drop an empty list down before a data set is selected.	Drop down list does drop down as empty before data set has been selected.	Dropdown list opens once clicked.	🟩
5	Dropdown list is updated to contain the iPhoneUID’s from the dataset	Once the dataset has been selected the dropdown list updates to show the iPhoneUID’s once the CSV file has been read.	iPhoneUIDs populate the drop down list when clicked after dataset has been selected.	Dropdown list displays all of the iPhoneUIDs which are contained within the dataset.	🟩
6	Cascade sub menu displays options in a cascade.	When clicked the sub menu should drop down options to be selected.	Sub menu drops down options to be clicked on.	Options appear under the option selected.	🟩
7	‘Exit’ option in the cascade menu exits program.	When the File and Exit option is pressed it should terminate the program.	Program terminates once option is clicked.	Program terminates.	🟩
8	‘About Us’ displays a message box containing text.	Message box appears once the About Us is pressed.	Message box appears containing text.	Message box containing text appears.	🟩
9	‘User Guide’ displays a user guide PDF when clicked by the user.	Once clicked the User Guide opens up in an internet browser.	User guide opens as a PDF in a browser window.	PDF will open in a browser window.	🟩

10	Correct UID is input into the program when selected from dropdown.	When clicked on from the drop down, the program will pass over the UID value.	IDE Displays the UID that has been selected as well as it is shown in the dropdown menu	In the IDE it will say “the user has selected [UID selected]”	See appendix 1.2
11	Program plots points on a map file.	Once coordinates, dataset and map are input then it will plot points.	Points are plotted.	Points are plotted.	
12	Plotting line is blue.	Line is set to 0,0,255 on the RGB scale.	Plotting line is blue once points have been plotted.	Blue line appears when plotting.	
13	Program plots points of UID: A841f74e620f74	The application will use the data that has been extracted from the CSV file to plot out the coordinates and points on the map image and then exports the image.	Plots the data on this UID to a map and exports it.	Plots the points relating to this UID on a map.	See Appendix 1.3
14	Program plots points of UID: 22223276ea84b	The application will use the data that has been extracted from the CSV file to plot out the coordinates and points on the map image and then exports the image.	Plots the data on this UID to a map and exports it.	Plots the points relating to this UID on a map.	See Appendix 1.4
15	Program plots points of UID: 510635002cb29	The application will use the data that has been extracted from the CSV file to plot out the coordinates and points on the map image and then exports the image.	Plots the data on this UID to a map and exports it.	Plots the points relating to this UID on a map.	See Appendix 1.5
16	Program plots points of UID: 7cbc37da05801	The application will use the data that has been extracted from the CSV file to plot out the coordinates and points on the map image and then exports the image.	Plots the data on this UID to a map and exports it.	Plots the points relating to this UID on a map.	See Appendix 1.6

17	Program plots points of UID: 7023889b4439d	The application will use the data that has been extracted from the CSV file to plot out the coordinates and points on the map image and then exports the image.	Plots the data on this UID to a map and exports it.	Plots the points relating to this UID on a map.	See Appendix 1.7
18	Program plots points of UID: 8425a81da55ec	The application will use the data that has been extracted from the CSV file to plot out the coordinates and points on the map image and then exports the image.	Plots the data on this UID to a map and exports it.	Plots the points relating to this UID on a map.	See Appendix 1.8
19	Program plots points of UID: 6882f6cf8c72d6	The application will use the data that has been extracted from the CSV file to plot out the coordinates and points on the map image and then exports the image.	Plots the data on this UID to a map and exports it.	Plots the points relating to this UID on a map.	See Appendix 1.9
20	Program plots points of UID: 1e33db5d2be36	The application will use the data that has been extracted from the CSV file to plot out the coordinates and points on the map image and then exports the image.	Plots the data on this UID to a map and exports it.	Plots the points relating to this UID on a map.	See Appendix 1.10
21	Program plots points of UID: 892d2c3aae6e5	The application will use the data that has been extracted from the CSV file to plot out the coordinates and points on the map image and then exports the image.	Plots the data on this UID to a map and exports it.	Plots the points relating to this UID on a map.	See Appendix 1.11
22	Visualisation of data is saved in program file under name “visualisation”	Saves in the plotted map as a PNG named Visualisation.	PNG File exported.	PNG file gets exported under the correct name.	

Appendix 1.2



```
C:\Users\AlexT\AppData\Local\Programs\Python\Python38-32\p
the user chose value a841f74e620f74ec443b7a25d7569545
```

Welcome to the Personal Security Application

a841f74e620f74ec443b7a25d7569545

Top Left Latitude: 0.0

Top Left Longitude: 0.0

Bottom Right Latitude: 0.0

Bottom Right Longitude: 0.0

Figure 25 Test 10 Results of UID being uploaded into program

Appendix 1.3

Note: All testing used the same Map excluding test 15 due to small area of operation.

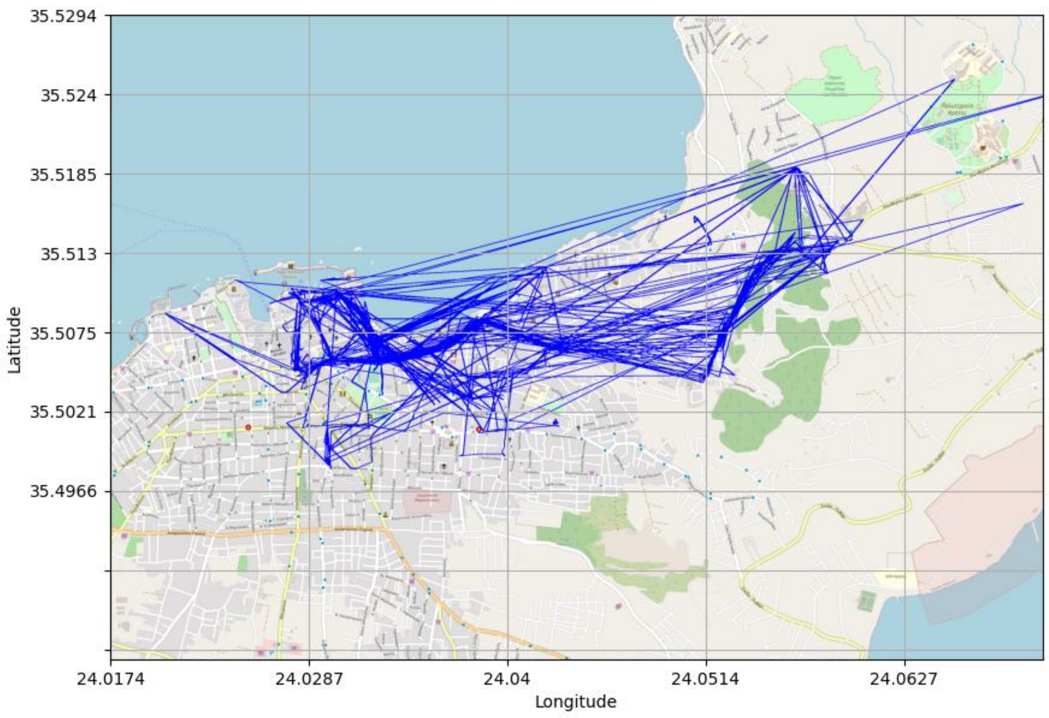


Figure 26 Result of test 13 on UID a841f74e620f74ec443b7a25d7569545

Appendix 1.4

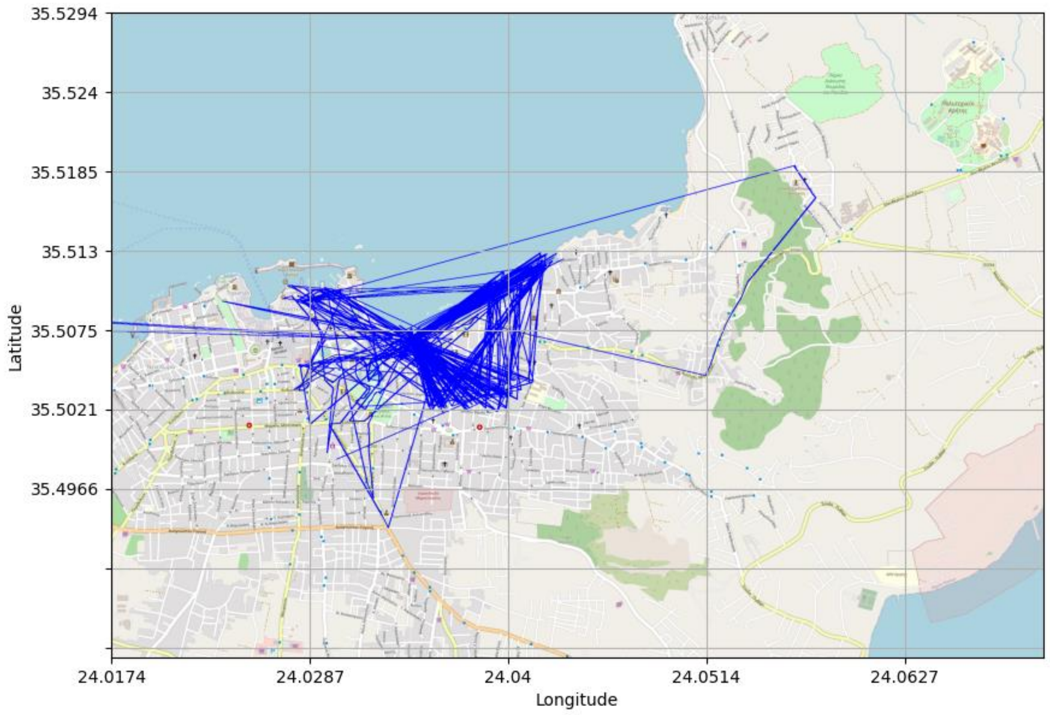


Figure 27 Result of test 14 on UID 22223276ea84bbce3a62073c164391fd

Appendix 1.5

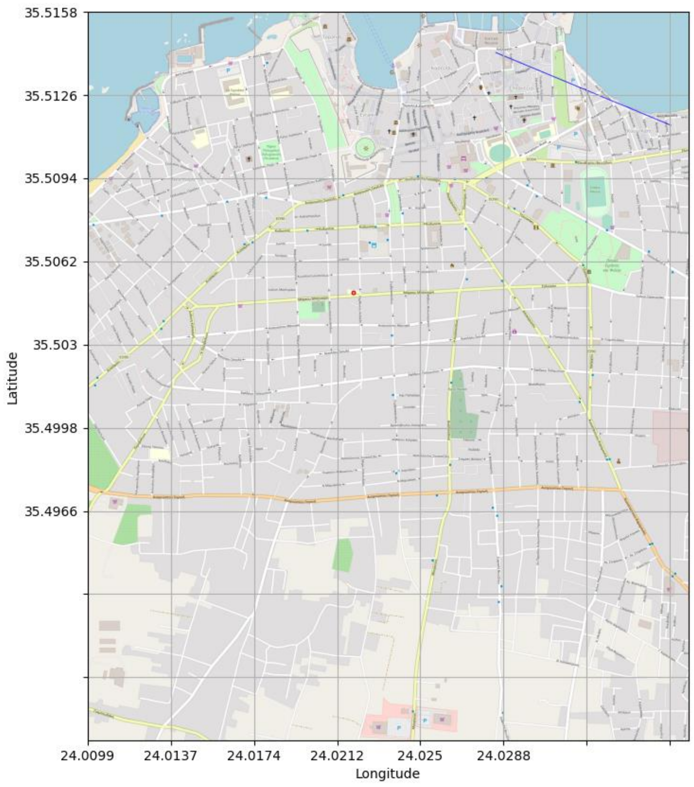


Figure 28 Result of test 15 on UID 510635002cb29804d54bff664cab52be

Appendix 1.6

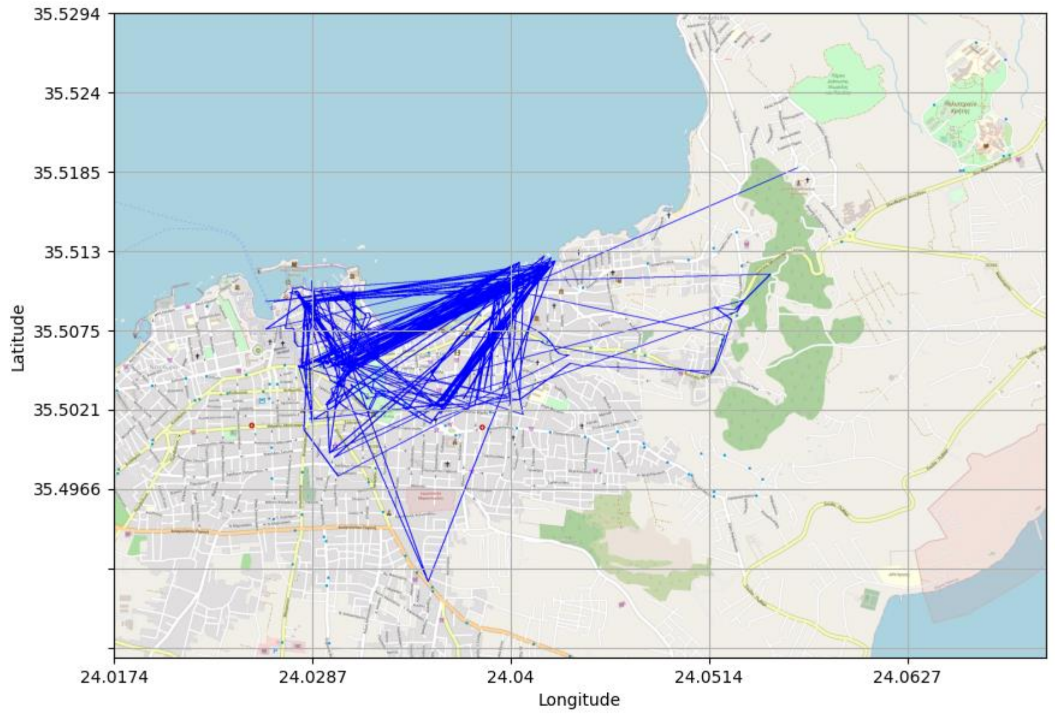


Figure 29 Result of test 16 on UID 7cbc37da05801d46e7d80c3b99fd5adb

Appendix 1.7

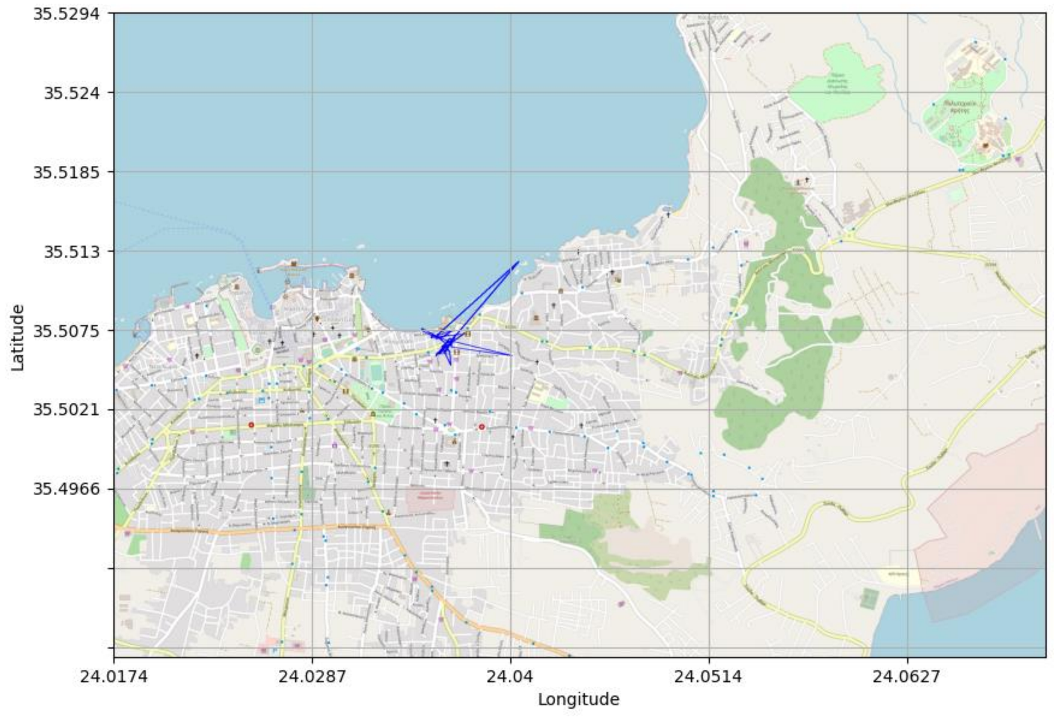


Figure 30 Result of test 17 on UID 7023889b4439d2c02977ba152d6f4c6e

Appendix 1.8

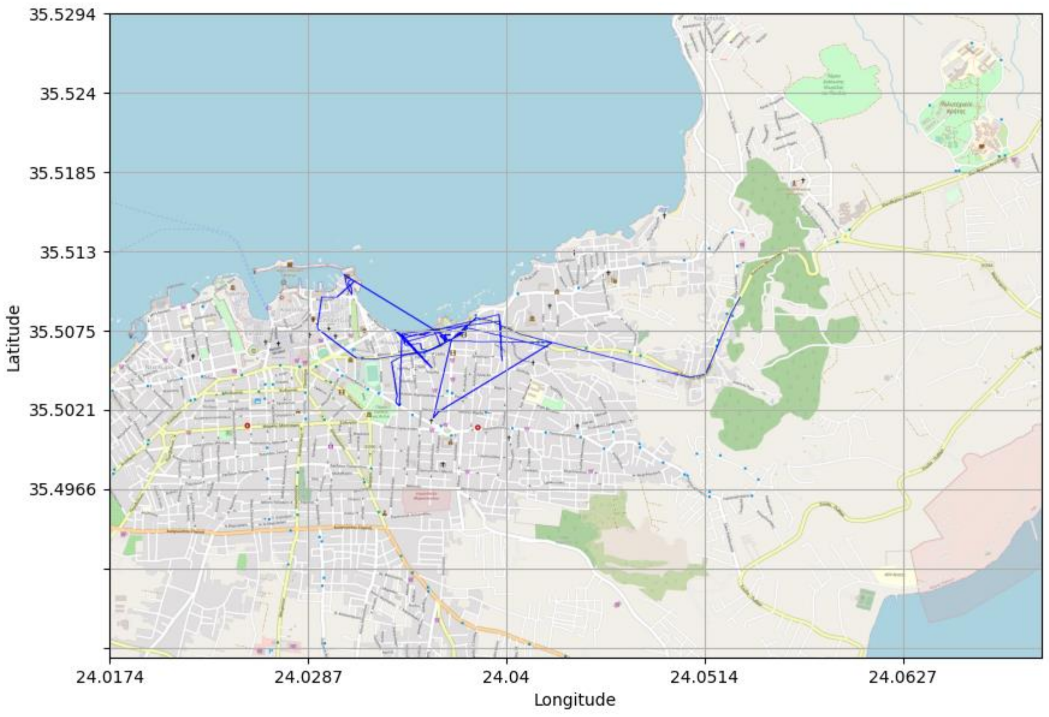


Figure 31 Result of test 18 on UID 8425a81da55ec16b7f9f80c139c235a2

Appendix 1.9

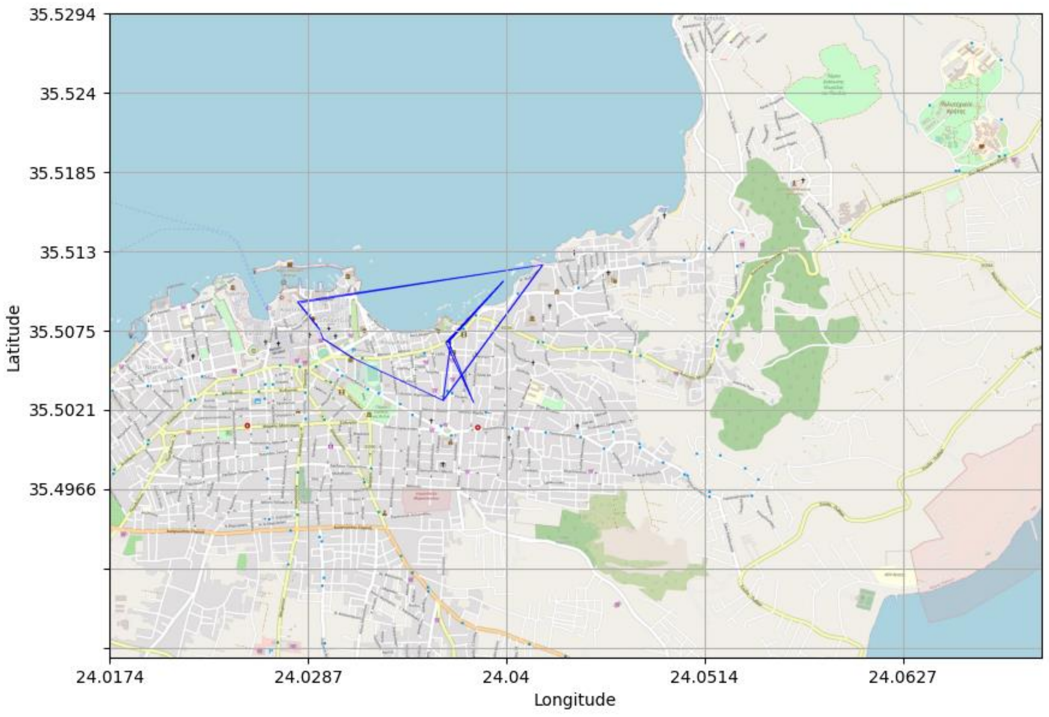


Figure 32 Result of test 19 on UID 6882f6cf8c72d6324ba7e6bb42c9c7c2

Appendix 1.10

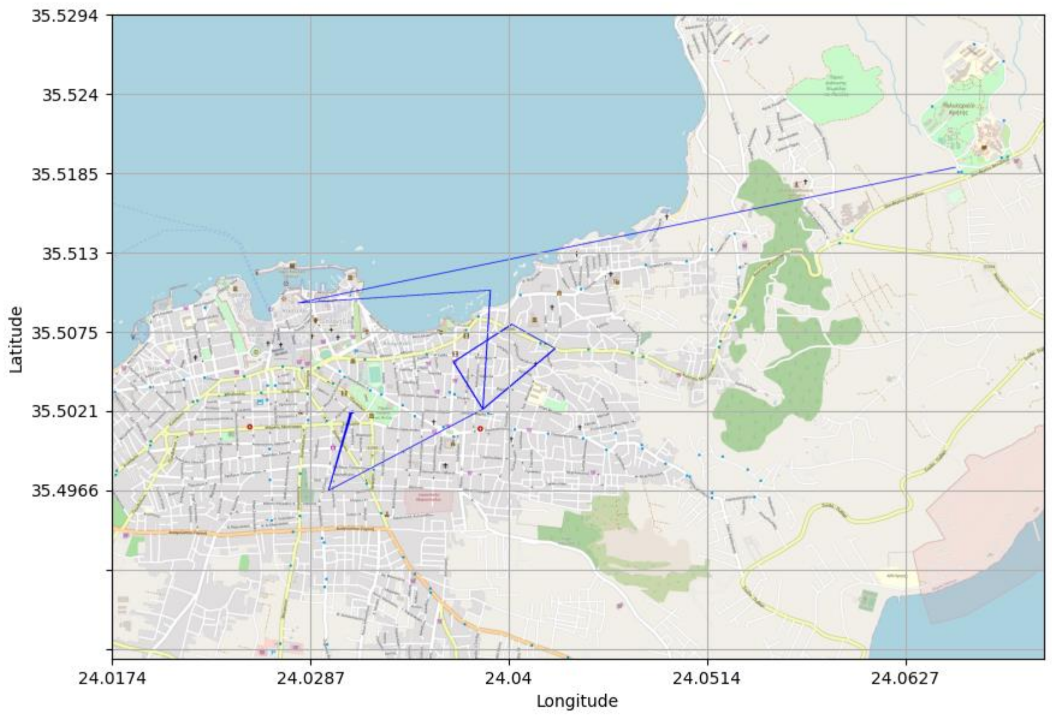


Figure 33 Result of test 20 on UID 1e33db5d2be36268b944359fbdbdad21

Appendix 1.11

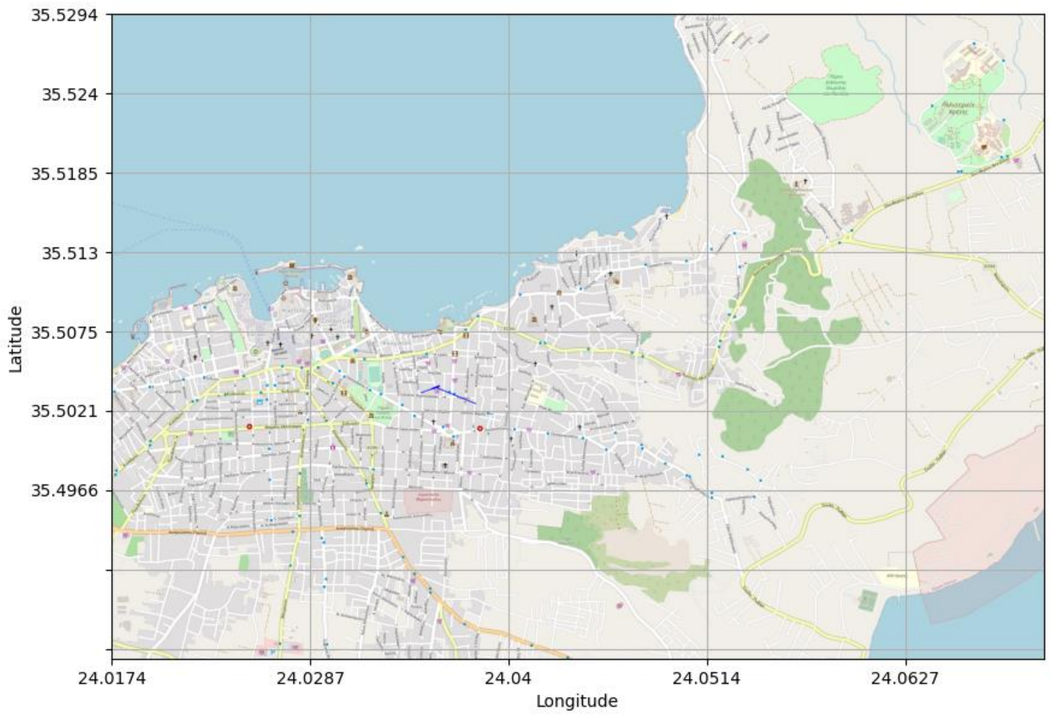


Figure 34 Result of test 21 on UID 892d2c3aae6e51f23bf8666c2314b52f