Thieft Technical Guide

A Technical Guide to Thieft - Bluetooth-Enabled Theft and Collision Detection Device for Vehicles.

Authors: Jack O'Reilly (18465312), Niall Bermingham (18392656)
Supervisor: Michael Scriney

Table of Contents

1. Introduction

1.1 Overview

Thieft is a bluetooth-enabled, theft and collision detection device for vehicles that can alert the user upon detection of a possible theft or collision incident. The device incorporates a Raspberry Pi, Accelerometer, GPS and SIM module. The Raspberry Pi single board computer has built-in bluetooth connectivity, which allows us to detect whether or not the vehicle owner's phone (or bluetooth device of their choosing) is within range of our device, which is stored inside their vehicle. The data collected in real-time from the GPS and Accelerometer is cached, and used to calculate the great circle distance using

the haversine formula. This enables us to detect displacement accurately and make decisions based on this data. If the vehicle is moving, and the user's authenticated device is not within range of the vehicle, they will be alerted to a possible theft incident via SMS. The user interface for our device allows users to manage their authenticated devices, provide details needed for alerting and tailor the device settings to their needs. Thieft API allows authorised users/third-party providers to view the data collected from possible theft and collision incidents.

We believe our device could be marketed towards insurance companies, who could offer discounted premiums to clients willing to install our device. In return the insurance company could benefit from:

1. Having access to theft and collision incident data via Thieft API.
2. Clients may drive safer while aware that their driving is being monitored.
3. Clients can alert emergency services quicker in the event of a theft.

Which we believe would result in less insurance claims being made, and less payments by the insurance company as a result.

1.2 Motivation

Both Niall and I have an interest in vehicles and security. Having both interned at large companies within the automotive industry (Jaguar Land Rover & General Motors) this past Summer, we wanted to build on the knowledge we acquired during our internships and challenge ourselves with a project that is both related to our interests, and one that also offers a wide range of areas to learn about while we develop and test our application. We began researching about the options that are currently out there for people looking to prevent vehicle theft, and how some of these methods are being exploited. Whilst various improvements in vehicle technology have certainly made life easier for

drivers, criminals have been able to come up with new ways of gaining access to vehicles.

For example, many relatively new vehicles have keys that send a short-range signal that tells them to unlock, even if they are in your pocket. Thieves exploit this technology using electronic car key relay boxes, putting one as close to your home as they can to receive signals coming from your car key fob, then boosting the signal to the second device near your car, tricking the car into thinking the key is nearby, which then unlocks the doors.
While there are several preventative measures available to take against vehicle theft (key fob blockers, steering locks, loud alarm etc.), we believe that being alerted to a possible theft incident via text message is certainly a cost-effective solution to this important problem. We were able to implement theft detection ahead of schedule, and wanted to make our device more useful to our target market: insurance companies. We decided to implement collision detection as a result, and also change how we store and query data to allow for API use.

2. Research

Prior to this project, we both had no experience working with bluetooth, accelerometer, GPS and SIM modules, so we began our research by first learning about these modules, and finding out if it would be possible to integrate them with a Raspberry Pi. Once we had agreed that it should be possible to work with these modules, we researched each of them in depth. Here is some of the research we carried out regarding each module:

2.1 AT Commands for the SIM Module

AT (Attention) commands are used to control MODEM's. In our project we are making use of the SIM7006E-hat modem, there are many services AT commands can provide us however within in the scope of our project we are mainly focusing on the GSM(Global System for Mobile Communications)/GPRS

(General Packet Radio Service) and the GPS (Global Positioning System) commands and their relevance to sending SMS messaging, connecting to the internet and for retrieving the latitude and longitude of our device. For the purpose of this project we have grouped the relevant AT commands for their respective operation.

2.2 GSM/GPRS Modem

The sms section of our SIM7006E consisted of regulating power to the device, The sms is known to drain power on text message so sorting out a modem that incorporated a capacitor of the recommended 5V was a necessity for our project. The initial modem we gathered for the project did not have an on board capacitor so extra steps had to be taken for the final module. Another area of research into the SMS section of this project was the different AT modes that the device could be set to. In our case the command “AT+CMGF” needed to be set before any message could be sent.

2.3 GPS Modem

Initially our project consisted of a separate GPS tracking system which after wiring and configuration showed how consistent and reliable this modem can be. This module took less power than the current modem. The benefit that was brought in due to the SIM7006E modem was its inclusion of an on board GPS modem. This brought flexibility to our device with this new integration we had access to AT commands which were previously unavailable on the previous module. Similar to the GSM/GPRS section we needed to set the modem to allow GPS data to be passed through “AT+CGPS=1,1”. With an extra command which is not outlined in our code which is the “AT+CGPSAUTO=1”. This At command sets up the modem to automatically gather gps information on startup of the device prior to setting the modem mode to gather gps information. Parsing the information gathered was a major step as it involved learning how gps coordinates are gathered and with implementations of this module varying from modem to modem, This is dependent on the implementation AT commands for said modem. This was another area of

research which was needed before a proper implementation could be completed.

2.4 DialUp

Our initial implemented a basic post system using AT commands within the Modem. However due to security concerns with creating BLOBS and sending the information securely and to the correct URL we opted to change our approach to using this optional setting. We incorporated a PPP (Point to Point Protocol) using the ppp open source program and created custom chatscripts for the devices communication with the server. This allowed us to bypass creating post requests from scratch and worrying about the security concerns by using python's built-in request library. We run the chatscripts using a custom ppp config file which we have called rnet for this project. We followed multiple references while creating this such as;
https://www.embeddedpi.com/documentation/3g-4g-modems/mypi-industrial-raspberry-pi-connecting-to-the-internet
https://docs.sixfab.com/page/setting-up-the-ppp-connection-for-sixfab-shield-hat

2.5 Accelerometer

An accelerometer is a sensor that measures the acceleration in a 3 dimensional plane. When researching this module it was important how we would use the data gatherers from the accelerometer to calculate the displacement within a given time. The accelerometer will not give us a speed but can be used for detecting rotation and displacement in a given time. For the purposes of our project we limited the time to 0.5 seconds however a more accurate implementation could use a shorter timeframe if need be, demonstrated by the below formula:

$$((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)^{1/2}$$

The accelerometer gives us information in bytes which we then translated to readable code, during research of this we found the python library for mpu6050 which returns a dictionary of the x, y and z coordinates which us determining the serial port and port address, in our case is “0x68”. We found this module to be vital to our implementation as it allowed us to determine if the device is moving before retrieving the GPS and sending SMS messages to the user.

2.6 Bluetooth

Our initial goal was for the device to detect a user's phone or other bluetooth device to be within range; this along with the accelerometer could allow us to detect if the registered user is within range of the device and if the device is in motion. As we are using a Raspberry Pi for our project we used the pyBluez library which allowed us to incorporate all the modules into one language.

2.7 Django

We determined that Django was the best option for our server backend due to its security features, speed, testing and as django incorporates a sqlite database within the django app installation we were able to make full use of its features for both all our backend needs such as user login, device registration, journey registration, etc.

2.8 React JS

React is a frontend JavaScript library which incorporates many custom objects and implements the ability to create static html files. React is a fantastic library and has many tutorials with django and with the amount of documentation and flexibility it provided for design it was a reasonable choice for this project. There are also many great libraries and frameworks that aid developers in

designing their user interfaces that are compatible with React, such as Material UI. Material UI is an open-source, front-end framework for React components. It is built using Less. Less (stands for Leaner Style Sheets), is a backward-compatible language extension for CSS. This allowed us to create a UI that is very user friendly and

3. Design

3.1 Case Design

As this is a physical device we decided to create a case to hold all of our hardware modules. The tools we used for this are Fusion360 and Ultimaker Cura. Fusion360 is a CAD (Computer Aided Design) software that is free for students and for personal projects. Ultimaker Cura is a slicing tool that takes in 3d models and slices them, layer by layer so a machine can either cut or 3d print the device. We chose to 3d print our case as the materials are cheaper than cutting and are quick to print with our largest design only taking 9 hours to print.

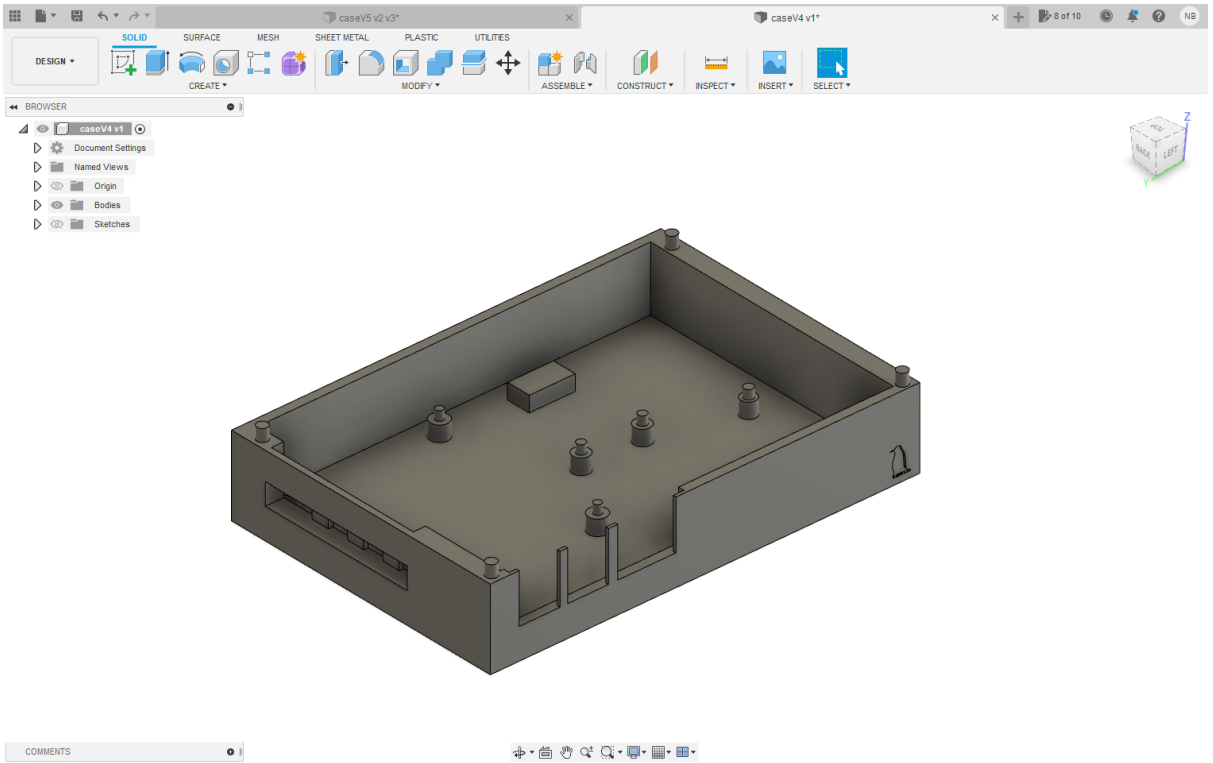


Fig 1. Original Case Design

Our initial design was created for the original set of hardware modules which required space for wiring and as you can see, only used pin holes to hold the modules in place. This is drastically different to our final design as with the new module we were able to shrink this design by over half the size and lifted the top by a few centimetres to account for the Raspberry Pi Hat.

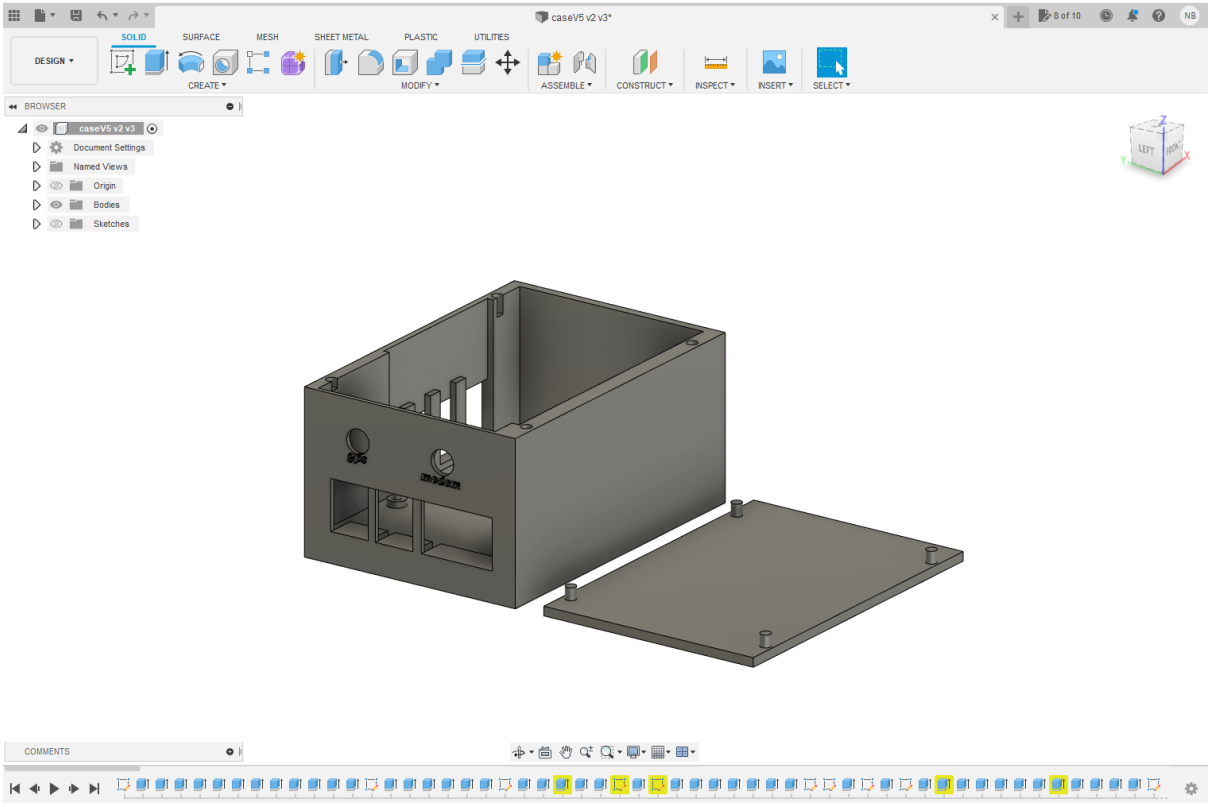


Fig 2. Final Case Design

The below image is the printed version of our final design, this stage consisted of many different iterations and designs before we were finally happy with the end result. This device would ideally be situated behind the dashboard of a vehicle however for demonstration purposes we feel this is a prototype of a design more than a finished product.

Fig 3. Image of final prototype

3.2 High-Level Design

High Level design is the architect that would be used for developing a software product. The diagrams below describe each component in our system and how they interact with each other.

3.2.1 Communication between components

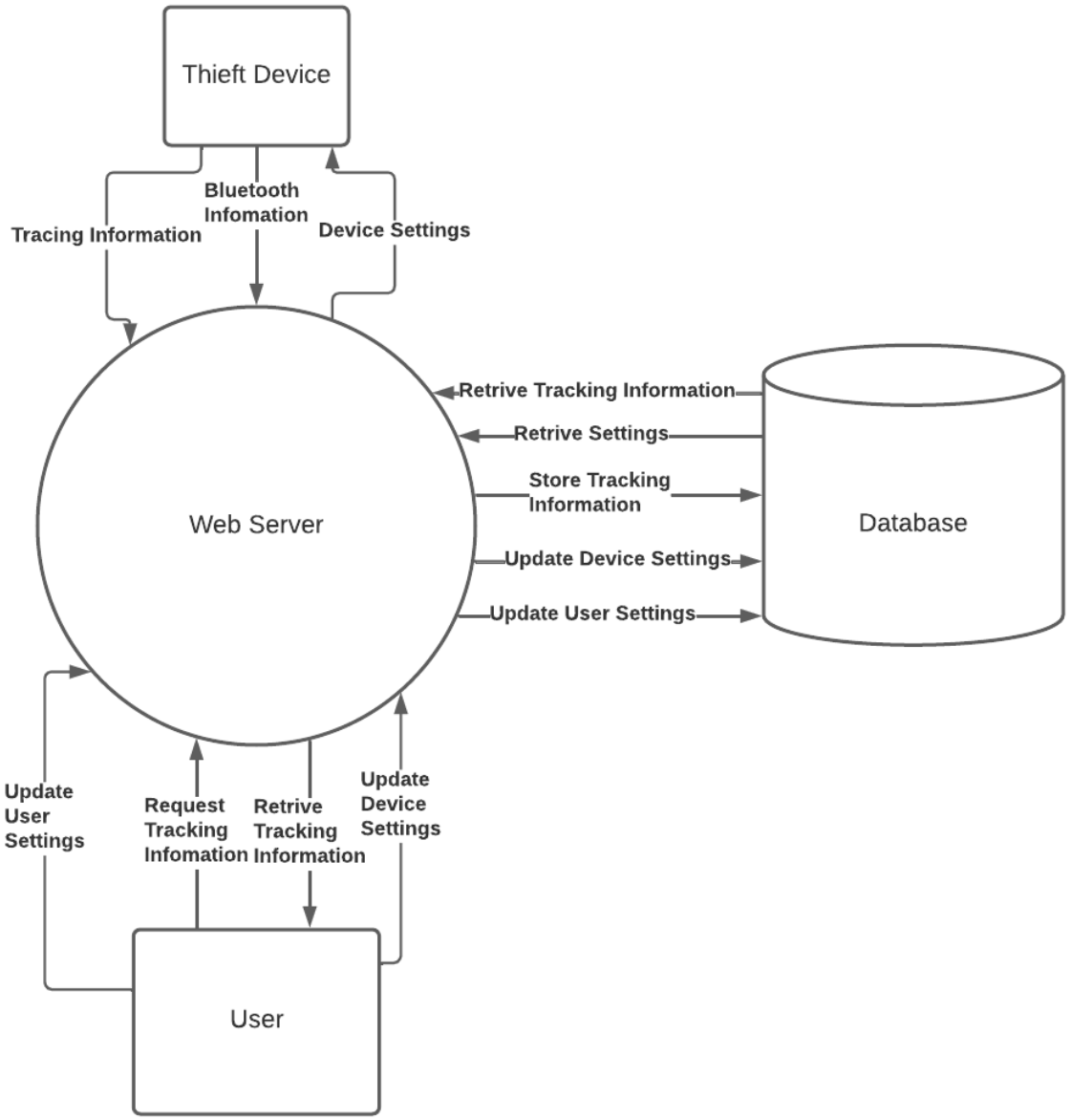


Fig 4. Communication diagram

3.2.2 User State Diagram

This state diagram represents what occurs when a user connects to the server.

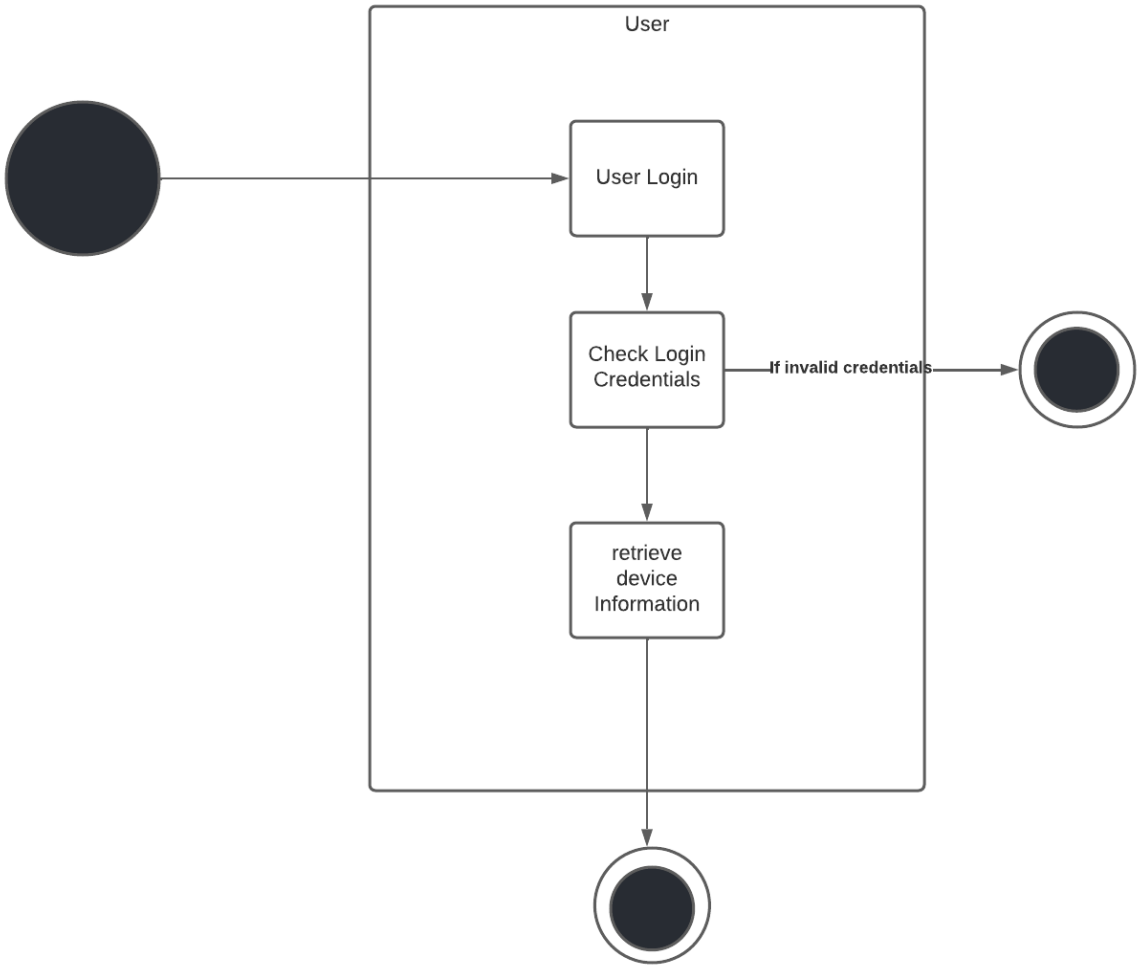


fig 5. User state diagram

3.3 DataBase Diagrams

3.3.1 Purpose

The purpose of our database is to link devices to their corresponding user. The device will hold the majority of the information which can only be accessed by an authorised user. User details consist of their username, password and their profile. The Profile holds their first name, last name, a unique id, address, city, country this information is not necessary for our project. However, we included

this information to make it easier to identify a user if the information collected was to identify a vehicle's owner. The phone number is necessary for our device to send SMS messages to an authorised user in case of a car theft. The device tables hold all the tracking information such as registered/authorised bluetooth devices that are allowed to operate the vehicle, the journey information so each individual journey can be logged and tracked, the tracking information which holds the latitude, longitude, accelerometer displacement and the GCD which can also be read as kilometres per journey interval e.g 2km per 10seconds. This information can be accessed by the user of a device through the UI features on the website.

3.3.2 Relationships diagrams

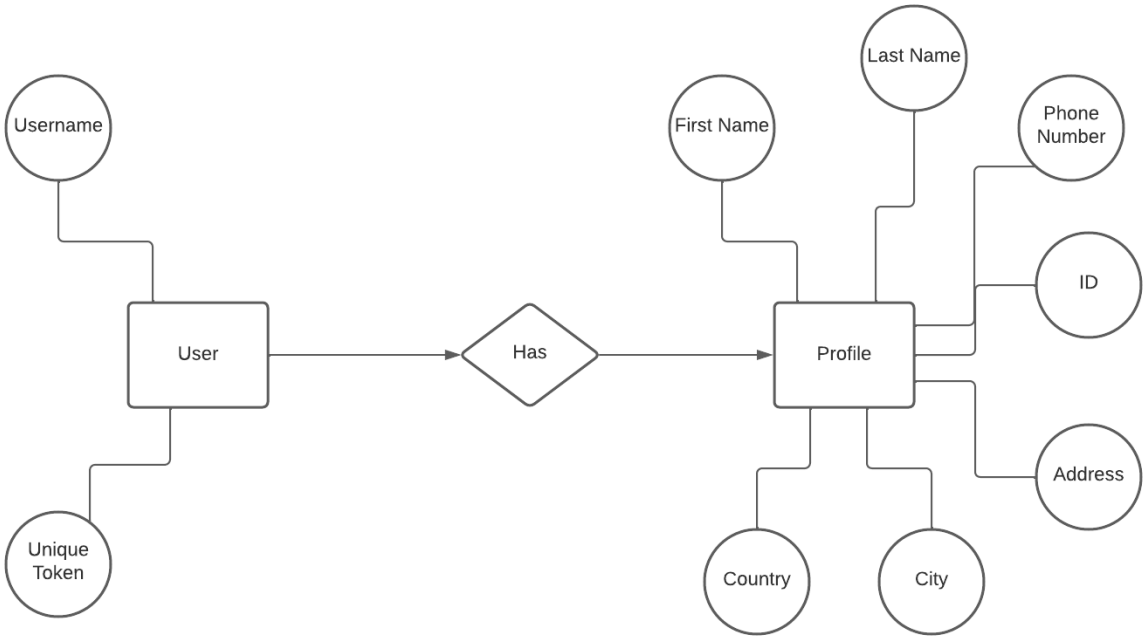


fig 6. User relationship diagram

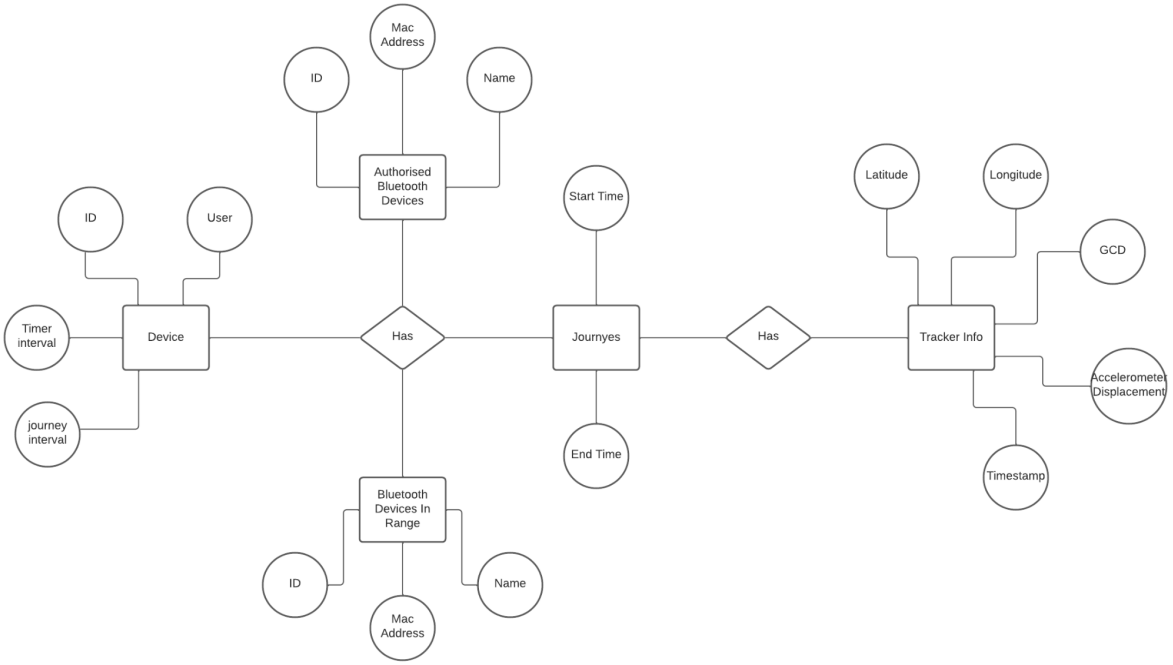


fig 7. Device relationship diagram

4. Implementation

4.1 Workflow

We worked using a continuous integration workflow, following the agile and scrum methodologies. Every week since getting approval of the project we would meet up and assign tasks that are to be done by the end of a given time period, this ranged from one to two weeks where we would review each other's code, discuss new ideas/implementations of our work before merging our code to the dev branch. We simulated different work environments for our project using git with the two main branches being master and dev, with sub branches named after each specific task which were developed, tested and then merged into dev. We aimed to use a system similar to “patches” where after a number of weeks had passed and each new task was peer reviewed we would merge into master and continue to work using the development branch as the main working branch. Below are a list of components we have

implemented in this project, along with their respective requirements and the reasoning behind our decision to use the different languages/frameworks we have used in this project.

4.2 Authenticate Bluetooth Devices

Authentication of Bluetooth devices is done on the web Application, a user can choose a device that is in range of the raspberry pi to authorise, which will add the device to a list of devices that is searched for when vehicle displacement is detected. That information is stored and sent to the raspberry pi and used as part of the theft detection algorithm. We used pybluez for our bluetooth needs as it allowed us to easily search for bluetooth devices in range, and integrated seamlessly with the Raspberry Pi.

```
import bluetooth as ble
import threading

class BluetoothPairing(object):
    def __init__(self, authorizedMacAddress):
        self.authorizedMacAddress = authorizedMacAddress
        self.bleCounter = 0
        self.devicesInRange = []
        self.running = False
        self.foundDevice = False

    def searchForBle(self):
        while (self.running):
            self.devicesInRange = ble.discover_devices(duration=5, flush_cache=True, lookup_names = True, lookup_class = True)
            self.bleCounter += 1
            for device in self.devicesInRange:
                if (any(item in self.authorizedMacAddress for item in device)):
                    self.foundDevice = True
                    self.bleCounter = 0
                    break


    def start(self):
        self.running = True
        findDevice = threading.Thread(target=self.searchForBle)
        findDevice.start()

    def kill(self):
        self.running = False
```

4.3 Adding Thieft Devices

Adding Thieft devices is carried out by a user where they enter the ID if a device of the corresponding ID has been found it will authorise and link the device to the user. This is done through the devices page on the UI to make this easier for multiple devices to be registered to a single user. The reasoning behind this is that we wanted to account for the user having multiple devices in multiple vehicles.

```
@csrf_exempt
def addDevice(request):
    if request.method == 'POST':
        postData = json.loads(request.body.decode('utf-8').replace("'", '"'))
        user = User.objects.get(token=postData['token'])
        device = DeviceSettings(
            user=user,
            texting_timeout=postData['texting_timeout'],
            update_journey_interval=postData['update_journey_interval']
        )
        device.save()
        return HttpResponse(device.id)
    return HttpResponse()

@csrf_exempt
def registerDevice(request):
    if request.method == 'POST':
        postData = json.loads(request.body.decode('utf-8').replace("'", '"'))
        if DeviceSettings.objects.filter(id=postData['device_id']).exists():
            device = DeviceSettings.objects.get(id=postData['device_id'])
            device.user = User.objects.get(token=postData['token'])
            device.save()
            return HttpResponse("OK")
        return HttpResponse("Error no device with id " + postData['device_id'])
    return HttpResponse()

@csrf_exempt
def getDevices(request):
    if request.method == 'POST':
        postData = json.loads(request.body.decode('utf-8').replace("'", '"'))
        user = User.objects.get(token=postData['token'])
        devices = DeviceSettings.objects.filter(user=user)
        devicedict = dict()
        for index, device in enumerate(devices):
            devicedict[index] = device.id
        return JsonResponse(devicedict)
    return HttpResponse()
```

4.5 Location Tracking

Location Tracking is done by the GNSS(Global Navigation Satellite System) module on the SIM7006E modem, Along with the AT commands. The location data is cached and can be viewed at any time by the user via the last location page on our application, and can also be queried using our API.

```python
@csrf_exempt
def getJourneyInfo(request, device_id, journey_id):
    if request.method == 'GET':
        device = DeviceSettings.objects.get(id=device_id)
        journey = device.journeys.get(id=journey_id)
        return HttpResponse(journey.trackings.all())
    return HttpResponse("Error method is not GET")

@csrf_exempt
def getAllJourneysForDevice(request):
    if request.method == 'POST':
        user = User.objects.get(token=json.loads(request.body.decode('utf-8'))["token"])
        devices = DeviceSettings.objects.filter(user=user)
        allJourneysVariable = dict()
        for device in devices:
            journey_info = dict()
            for j in device.journeys.all():
                journey_info[j.id] = [{'latitude': t.latitude, 'longitude': t.longitude, 'timestamp': t.timestamp, 'gcd': t.gcd,'accelerometer
            allJourneysVariable[device.id] = journey_info
        return JsonResponse(allJourneysVariable, safe=False)
    return HttpResponse("Error method is not GET")

@csrf_exempt
def createJourney(request):
    if request.method == 'POST':
        postData = json.loads(request.body.decode('utf-8').replace("'", '"'))
        device = DeviceSettings.objects.get(id=postData['device_id'])
        journey = Journey()
        journey.save()
        device.journeys.add(journey)
        device.save()
        return HttpResponse(journey.id)
    return HttpResponse("Error method is not POST")
```

```python
@csrf_exempt
def postJourneyInfo(request):
    if request.method == 'POST':
        postData = json.loads(request.body.decode('utf-8').replace("'", '"'))
        user = User.objects.get(username=postData['username'])
        device = DeviceSettings.objects.get(user=user, id=postData['device_id'])
        device_journeys = device.journeys.all()
        journey = device_journeys.get(id=postData['journey_id'])
        trackerType = "Theft" if postData['accelerometer_displacement'] < 60 else "Collision"
        journey.trackings.add(TrackerManager().create_tracker(lat=postData['latitude'], long=postData['longitude'], gcd=postData['gcd'],acce
        journey.save()
        device.save()
        return HttpResponse(str(journey))
    return HttpResponse("post Failed")

@csrf_exempt
def getjourneyLocation(request):
    if request.method == 'POST':
        postData = json.loads(request.body.decode('utf-8'))
        user = User.objects.get(token=postData['token'])
        device = DeviceSettings.objects.get(user=user, id=postData['device_id'])

        journey = device.journeys.get(id=postData['journey_id'])
        return JsonResponse(TrackerManager().get_journeyInfo_location(journey))
    return HttpResponse("Error method is not POST")
```

4.6 Theft Detection

Theft detection is done by using the accelerometer and bluetooth modules. We detect if a registered bluetooth device is in range of the Raspberry Pi. We measure the accelerometer displacement over a set time interval to detect motion and if the registered bluetooth device is not within range we then mark that as a possible car theft. Sending all the information such as bluetooth, latitude and longitude, gcd and the accelerometers displacement to the server which is then sorted and stored for the correct user. This is possible by the SIM7006E modem onboard the device allowing for internet connection and

gathering of this information. We also send a SMS message to the user stating that a possible car theft is in progress.

```
def haversine(lon1, lat1, lon2, lat2) -> float:
    """
    Calculate the great circle distance in kilometers between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
    # compute great circle distance using the haversine formula
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
    c = 2 * asin(sqrt(a))
    r = 6371 # radius of earth in kilometers
    return c * r

def sendMsg(msg: str, phoneNumber: str) -> None:
    sms = SMSModule()
    sms.sendSMS(message=msg, phoneNumber=phoneNumber)
    return None

def getGPS() -> dict:
    gps = GPSModule()
    return gps.get_gps_position()

def calcDisplacement():
    sensor = mpu6050(0x68)
    val1 = sensor.get_accel_data()
    time.sleep(0.5)
    val2 = sensor.get_accel_data()
    return sqrt((val1['x'] - val2['x'])**2 + (val1['y'] - val2['y'])**2 + (val1['z'] - val2['z'])**2)
```

```
def main() -> None:
    settings = Settings()
    settings.openSettings()
    bleModule = BluetoothPairing(settings.macAddress)
    bleModule.start()
    module = SimModule("SIM Module")
    module.power_on()
    ppp = PPP()


    while ((displacement := calcDisplacement()) < 5 ):
        if (not bleModule.foundDevice and bleModule.bleCounter > 5):
            break
        print(bleModule.bleCounter)
        print(displacement)
        print("waiting for crash")
    bleModule.kill()
    sms = SMSModule()
    sms.sendSMS(message="Car in Motion detected", phoneNumber=settings.phoneNumber)
    journey_id = settings.sendPost("tracking/data/createJourney/", dict(device_id=settings.device_id))
    try:

        while True:
            gps_val1 = getGPS()
            time.sleep(settings.update_journey_interval)
            gps_val2 = getGPS()
            gcd = haversine(gps_val1['longitude'], gps_val1['latitude'], gps_val2['longitude'], gps_val2['latitude'])
            displacement = calcDisplacement()
            settings.sendPost("tracking/data/post/", dict(
                device_id=settings.device_id, username=settings.related_user, journey_id=journey_id, latitude=gps_val2["latitude"], longitude=
                gcd=gcd, accelerometer_displacement=displacement

            ))
            time.sleep(settings.texting_timeout)
    except Exception as e:
        module.power_off()
        bleModule.kill()
        print(e)
```

4.7 Collision Detection

Collision detection is something that we decided to implement at a late stage of our project, in order to add more functionality and allow users to distinguish between different types of alerts. We made use of the haversine formula and compared the displacement values to the testing parameters we used to identify a collision. We lacked the necessary hardware components to implement this to a high standard, such as sensors and other hardware (such as a microphone, which could be used to detect the noise of the crash, air bag deployment etc.)

5. Problems and Solutions

5.1 AT Commands

A common problem we encountered was with our modem. This involved finding the correct AT(attention) commands. Many modems have a variety of

different At commands accessible to them. In our case we were able to distinguish the different commands available through the production companies website;
https://www.waveshare.com/wiki/SIM7600E-H_4G_HAT
As our modem contained specific commands for the GNSS, GSM and GPRS we had to tackle learning and understanding of each command. Throughout our research we often came across different implementations of these features which utilised newer commands which were not available with our modem and translating those features became a struggle especially when working with new hardware which we had not worked on before. Through the combination of reading documentation and using sites like stack Overflow we were able to conquer this issue and develop the basis of our GPS, SMS and internet connectivity with our project.

5.2 Wiring and voltage

Wiring and voltage became a great concern when we began our project, as in the beginning we were using separate modules for each functionality our final project was to utilise. We quickly discovered the basic voltage and amperage with our modules needed careful consideration before we were able to gather and use the data we had been collecting. We used basic wiring diagrams to keep track of our layout and to keep notes of how much voltage was needed for each component. This along with testing allowed for our first prototype to be created however after careful consideration we decided to bring in a new module which replaced the SIM and GPS modules greatly reducing the size of the device and made our initial wiring plan out-dated.

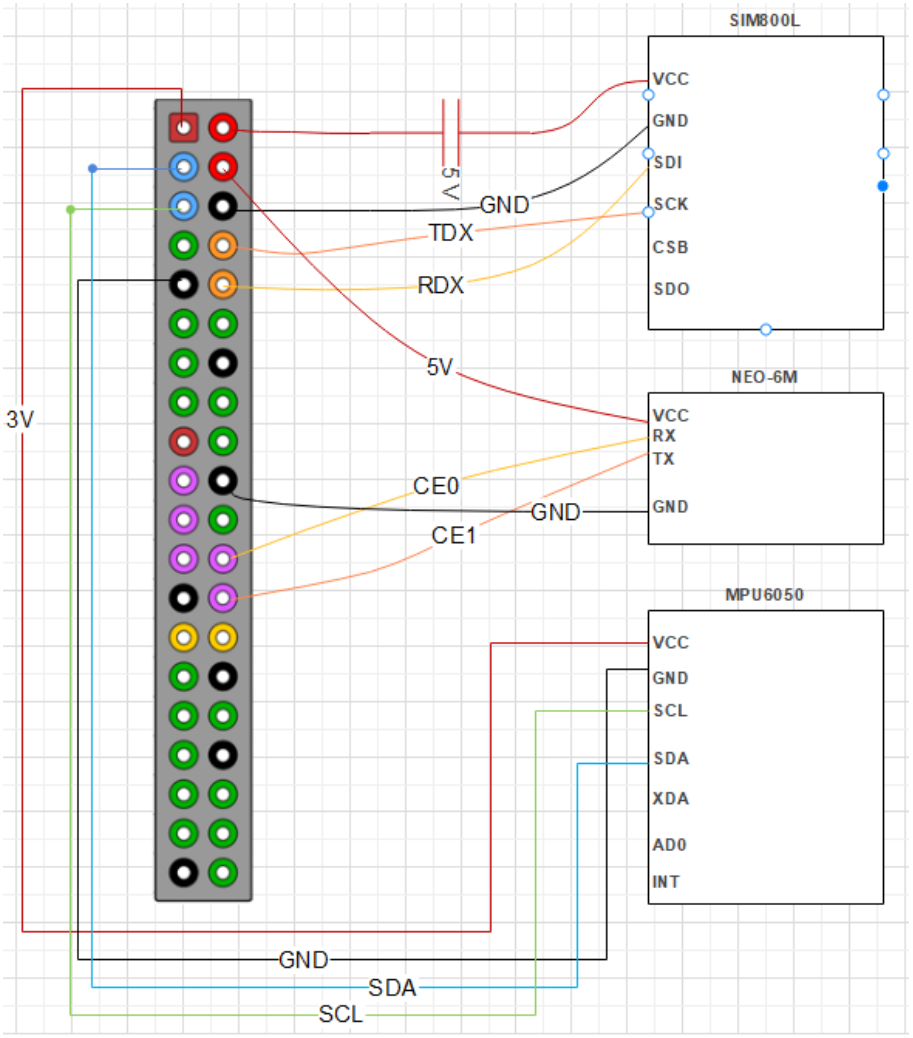


fig 8. Wiring Diagram

5.3 Case Design

Our process for designing the case was to create a safe way to transport our project. Although our project is mainly software orientated with the inclusion of hardware we found it difficult to manage the wiring and transportation of the device. This was also mandatory for our project as this device is intended to be installed into a vehicle and without any type of housing the wiring and hardware could become damaged if not properly protected. The problems with our design was to implement something that could hold a raspberry pi safely through the screw mounts which involved a deep dive into the raspberry pi schematic diagrams to gather the measurement for each mount point, schematics for each individual module before we upgraded our device to

include a newer module. Our case design went through several iterations before we came to its final result. The original design did not include the newer SIM modem and instead included the measurement for each individual module as seen below.

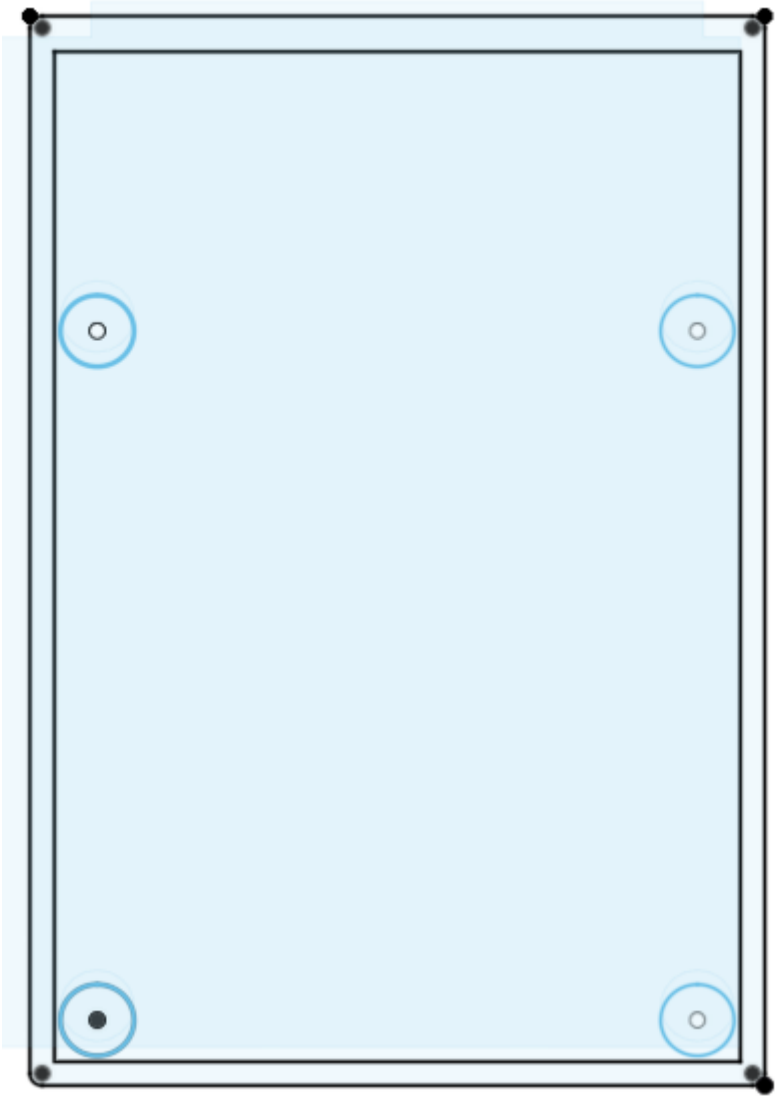


fig 9. Raspberry Pi sketch

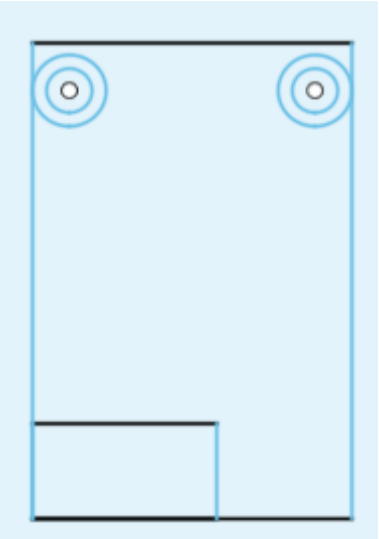


fig 10. GPS module sketch

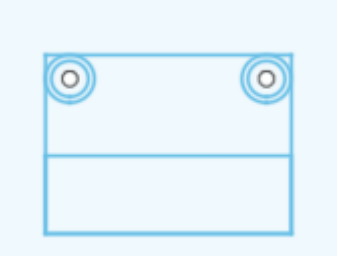


fig 11. Accelerometer sketch

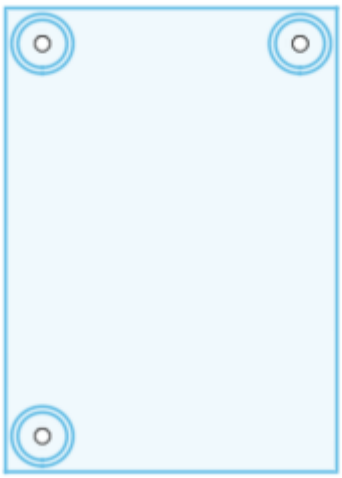


Fig 12. GSM/GPRS module sketch

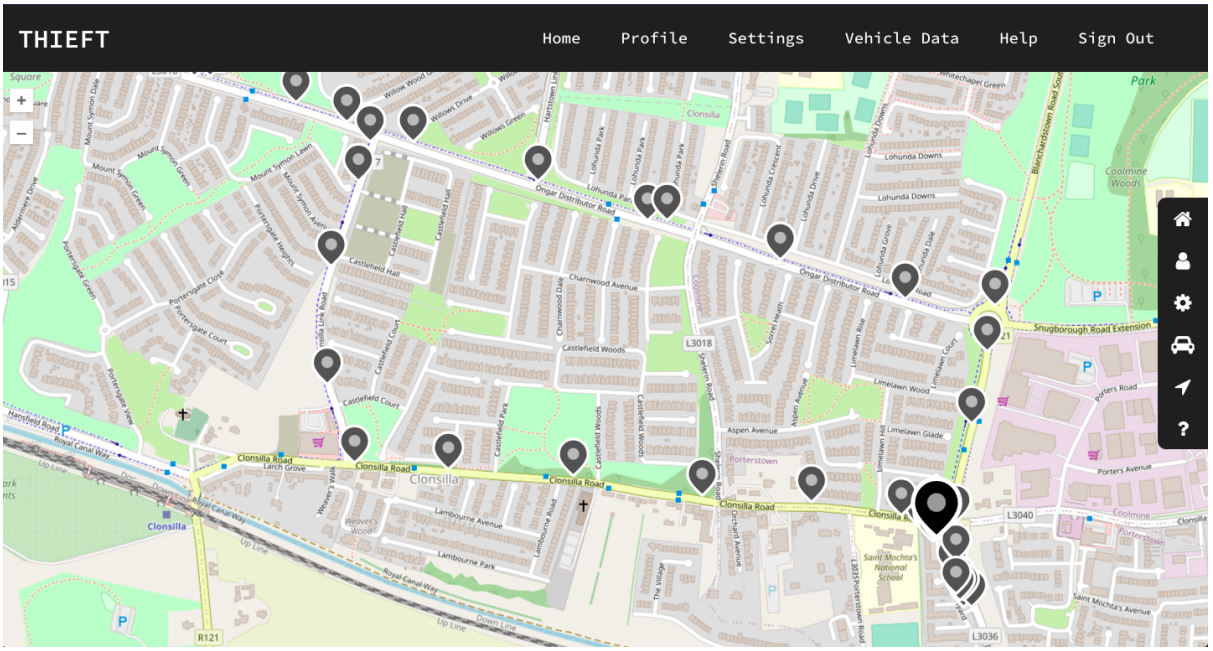
With these original sketches we were able to create and model a case design which can be seen in fig (fig number to original case) However with the introduction of a new modem the sketches for the individual modules became unused in the final design.

5.4 Django setup

Neither of us have used django in any of our studies or in work so this was a great hurdle to overcome. Throughout this project we had already studied and developed our skills in other aspects and the decision to choose django as our backend system was simply due to its security integration, database creation and as it was written in python, a language both of us are comfortable in writing. Django is vastly different from some of the other backend systems we are used to working in so it was a step up in learning the software and ideology. Models are a staple in Django allowing us to create SQLite database tables and storing that information using python's class system however as neither of us has had much experience in the library learning the interactions of this system proved to be difficult and had a large learning curve.

5.5 Mapbox Integration

When we made the decision to attempt to cache and display journey data on our application, we decided to integrate and use the Mapbox API for all of our map-related operations. The initial implementation was seamless, but as we added more complexity to our design, we ran into some problems. After spending a considerable amount of time attempting to solve a babel-loader related issue when attempting to place multiple markers on the map, we decided to look for alternative solutions. One of the alternatives, Pigeon Maps, was able to fulfil our needs regarding the ability to place multiple markers on the map. We decided to use this solution as there was very little difference between the two APIs, apart from the ease of use that Pigeon Maps seemed to offer compared to Mapbox.

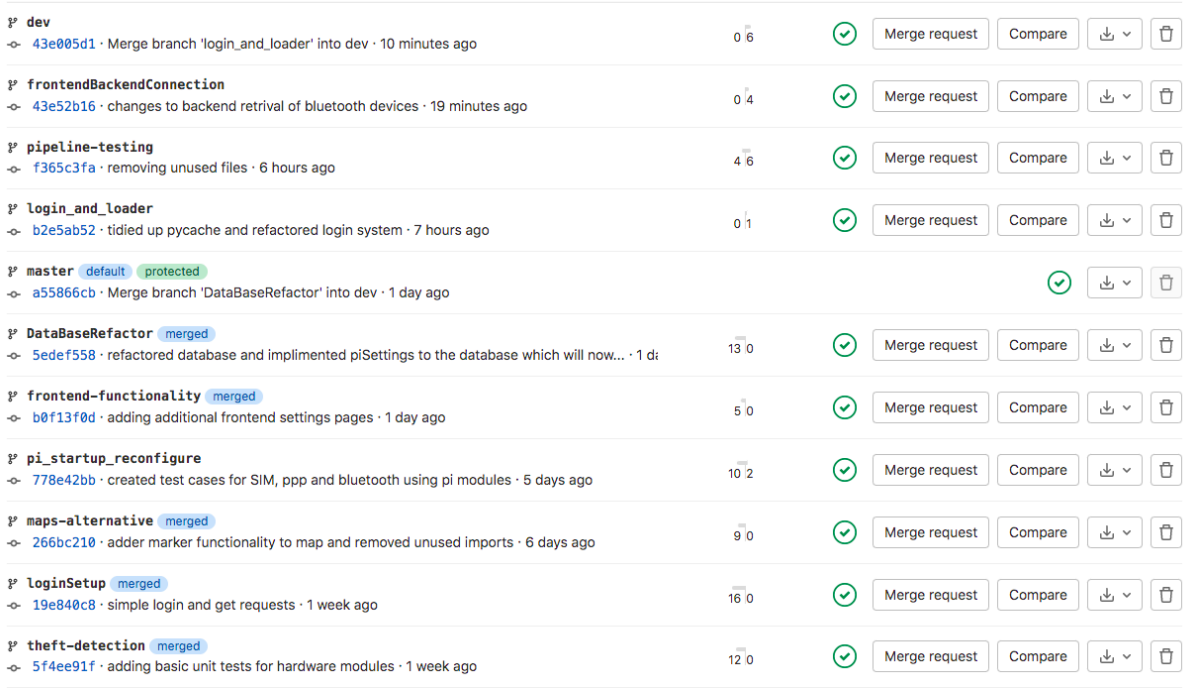


6. Testing

6.1 Git and CI/CD

Our Git flow branching strategy consisted of:

Creating new descriptively-named branches off the dev branch for new features, and then branching off those branches for .
Commiting new work to our local branches and regularly pushing our changes.
When our work is ready to merge to dev, we would open a pull request.



Our CI/CD pipeline that runs our migrations, integration and react tests:

```yaml
stages:
  - build
  - test

default:
  image: ubuntu:latest

before_script:
  # accept default answer for all questions while installing system updates and requirements
  - DEBIAN_FRONTEND=noninteractive apt-get update
  - apt -yq install apt-utils
  - apt -yq install python3 python3-pip
  - DEBIAN_FRONTEND=noninteractive apt install -yq npm nodejs
  - cd ./src/thieft && pip3 install -r requirements.txt

migrations:
  stage: build
  script:
    - echo "Creating and applying migrations..."
    - python3 manage.py makemigrations &&
      python3 manage.py migrate

integration:
  stage: test
  script:
    - echo "Running integration tests..."
    - python3 manage.py test

react:
  stage: test
  script:
    - cd ./frontend
    - echo "Running react tests..."
    - npm ci && npm test
```

6.2 Unit Testing

Unit testing had been a core part of our code with using the unit test library for the raspberry pi modules as well as the unit test integration with django.

6.2.1 Raspberry PI

The tests below are hardware dependent so testing these features proved to be challenging. As most of our tests return unknown variables such as the bluetooth, accelerometer, GPS we can only check the return types, the format of the functions and test that no errors occur either with their connectivity to the device which would be a hardware connection error such as incorrect wiring or that they are returning the expected return type.

6.2.1.1 PPP - Dial UP Internet

Due to the nature of our project we have written hardware specific code for ensuring the functionality of our modules. This includes the point to point access for the dial up feature of our modem where we ensure a stable connection is available before we send any information to the server. This involved setting up the chatscripts to allow connection to the APN(Access Point Name) in our case we are using a three sim so the three apn is used.

```
class TestPPP(unittest.TestCase):
    """
    Tests for ppp aka (internet connection on command)
    """
    def test_ppp_start_connection(self):
        """
        Test for starting a connection using ppp
        """
        ppp = PPP()
        self.assertTrue(ppp.startConnection())
        self.assertTrue(ppp.state)

    def test_ppp_end_connection(self):
        """
        Test for ending a connection using ppp
        """
        ppp = PPP()
        self.assertTrue(ppp.endConnection())
        self.assertFalse(ppp.state)

if __name__ == '__main__':
    unittest.main(TestPPP().test_ppp_start_connection())
    unittest.main(TestPPP().test_ppp_end_connection())
```

6.2.1.2 SIM7006E Testing

The Sim7006E is an oddity for our testing suites. As we are reading and writing bytes all of the tests on this module must be run one at a time. This is a limiting feature with the hardware and to not create bytes errors or allowing multiple features such as the GNSS, GSM or GPRS to create issues. These tests are for checking that the return type is what we expect and that they are not running at the same time.

```python
from ModulePackage.SIMModule import SimModule

class TestSimModule(unittest.TestCase):
    """
    Tests for Sim module on raspberry pi
    - warning due to the nature of sending information to different information to the pi gpio ensure each test is run individually for all modules
    """
    def test_power_on(self):
        """
        Test to ensure that the module has powered on
        """
        sim_module = SimModule()
        self.assertTrue(sim_module.power_on())

    def test_power_off(self):
        """
        Test to ensure that the module has powered off
        """
        sim_module = SimModule()
        self.assertTrue(sim_module.power_off())

    def test_at_command(self):
        """
        Test to ensure that the module has powered off
        """
        sim_module = SimModule()
        self.assertTrue(sim_module.power_on())
        # asserts command at is ok
        self.assertTrue(sim_module.send_at_command("AT", "OK", 2))
        # asserts command at is error expected failure
        self.assertFalse(sim_module.send_at_command("AT", "ERROR", 2))
        self.assertTrue(sim_module.power_off())

if __name__ == '__main__':
    unittest.main(TestSimModule().test_power_on())
    unittest.main(TestSimModule().test_power_off())
    unittest.main(TestSimModule().test_at_command())
```

6.2.1.3 Accelerometer Testing

As the accelerometer and bluetooth are not a part of the SIM7006E Modem we are able to run these tests at the same time as the SIM7006E modem tests. However the accelerometer returns unknown values after running and we cannot determine the answer beforehand we can only check for the module is connected to the right integrated circuit connectors as well as the return type.

6.2.1.4 Bluetooth Testing

Our bluetooth testing checks if a list of tuples is returned by the pyBluez library. We use this library as it integrates the scanning and connecting to devices within its library. This is an invaluable library for our project so testing ensures that this is interacting with the hardware correctly before use.

```python
class TestBluetooth(unittest.TestCase):
    """
    Tests for bluetooth module on raspberry pi
    """
    def test_scan(self):
        """
        Test needs at least one bluetooth device to be in range
        - this is to ensure that the device takes in a tuple for each bluetooth device
        """
        devicesInRange = bluetooth.discover_devices(lookup_names = True, lookup_class = True)
        self.assertEqual(type(devicesInRange), list)
        self.assertEqual(type(devicesInRange[0]), tuple)
```

6.2.2 Django Tests

Django's integration with python's unittest library allowed for easy implementation to check the database structure as well as the views and urls ensuring correct movement of information and that each view was only running depending on the request type e.g POST, GET, DELETE etc. Our tests mainly deal with the storing and retrieving of information from the database.

```python
class UserTestCase(TestCase):

    def setUp(self):
        """test for user object"""
        User.objects.create(username='testuser1', password='12345', token='example_token!')
        User.objects.create(username='testuser2', password='54321', token="right_token!")


    def testPassword(self):
        """test for user password"""
        user1 = User.objects.get(username='testuser1')
        user2 = User.objects.get(username='testuser2')

        self.assertEqual(user1.password, '12345')
        self.assertEqual(user2.password, '54321')


    def testToken(self):
        """test for user token"""
        user1 = User.objects.get(username='testuser1')
        user2 = User.objects.get(username='testuser2')

        self.assertEqual(user1.token, 'example_token!')
        self.assertNotEqual(user2.token, 'wrong_token!')
```

6.3 Integration Testing

Our integration tests were part of our CI Pipeline, which allowed us to run the tests together in a suite like environment. If any one of the tests failed, the pipeline would fail. We made use of the built in test functionality offered by Django when testing the models for each one of our different components, and also verified that the migrations were functional before testing these models.

```
3125 $ echo "Creating and applying migrations..."
3126 Creating and applying migrations...
3127 $ python3 manage.py makemigrations && python3 manage.py migrate && python3 manage.py check
3128 System check identified some issues:
3129 WARNINGS:
3130 ?: (staticfiles.W004) The directory '/builds/oreilj46/2022-ca400-oreilj46-berminn4/src/thieft/fronten
    d/build/static' in the STATICFILES_DIRS setting does not exist.
3131 Migrations for 'authentication':
3132   authentication/migrations/0006_alter_user_token.py
3133     - Alter field token on user
3134 Operations to perform:
3135   Apply all migrations: admin, auth, authentication, contenttypes, manage_device, sessions, tracker
3136 Running migrations:
3137   Applying contenttypes.0001_initial... OK
3138   Applying authentication.0001_initial... OK
3139   Applying admin.0001_initial... OK
3140   Applying admin.0002_logentry_remove_auto_add... OK
3141   Applying admin.0003_logentry_add_action_flag_choices... OK
3142   Applying contenttypes.0002_remove_content_type_name... OK
3143   Applying auth.0001_initial... OK
3144   Applying auth.0002_alter_permission_name_max_length... OK
3145   Applying auth.0003_alter_user_email_max_length... OK
3146   Applying auth.0004_alter_user_username_opts... OK
3147   Applying auth.0005_alter_user_last_login_null... OK
3148   Applying auth.0006_require_contenttypes_0002... OK
3149   Applying auth.0007_alter_validators_add_error_messages... OK
3150   Applying auth.0008_alter_user_username_max_length... OK
```

```
3150   Applying auth.0008_alter_user_username_max_length... OK
3151   Applying auth.0009_alter_user_last_name_max_length... OK
3152   Applying auth.0010_alter_group_name_max_length... OK
3153   Applying auth.0011_update_proxy_permissions... OK
3154   Applying auth.0012_alter_user_first_name_max_length... OK
3155   Applying authentication.0002_alter_user_token... OK
3156   Applying authentication.0003_alter_user_token... OK
3157   Applying authentication.0004_alter_user_token... OK
3158   Applying authentication.0005_alter_user_token... OK
3159   Applying authentication.0006_alter_user_token... OK
3160   Applying tracker.0001_initial... OK
3161   Applying manage_device.0001_initial... OK
3162   Applying sessions.0001_initial... OK
3163 System check identified some issues:
3164 WARNINGS:
3165 ?: (staticfiles.W004) The directory '/builds/oreilj46/2022-ca400-oreilj46-berminn4/src/thieft/fronten
    d/build/static' in the STATICFILES_DIRS setting does not exist.
3166 System check identified some issues:
3167 WARNINGS:
3168 ?: (staticfiles.W004) The directory '/builds/oreilj46/2022-ca400-oreilj46-berminn4/src/thieft/fronten
    d/build/static' in the STATICFILES_DIRS setting does not exist.
3169 System check identified 1 issue (0 silenced).
3174 Job succeeded
```

```
3163 $ echo "Running integration tests..."
3164 Running integration tests...
3165 $ python3 manage.py test
3166 Found 4 test(s).
3167 Creating test database for alias 'default'...
3168 System check identified some issues:
3169 WARNINGS:
3170 ?: (staticfiles.W004) The directory '/builds/oreilj46/2022-ca400-oreilj46-berminn4/src/thieft/fronten
    d/build/static' in the STATICFILES_DIRS setting does not exist.
3171 System check identified 1 issue (0 silenced).
3172 ....
3173 ----------------------------------------------------------------------
3174 Ran 4 tests in 0.024s
3175 OK
3176 Destroying test database for alias 'default'...
3181 Job succeeded
```

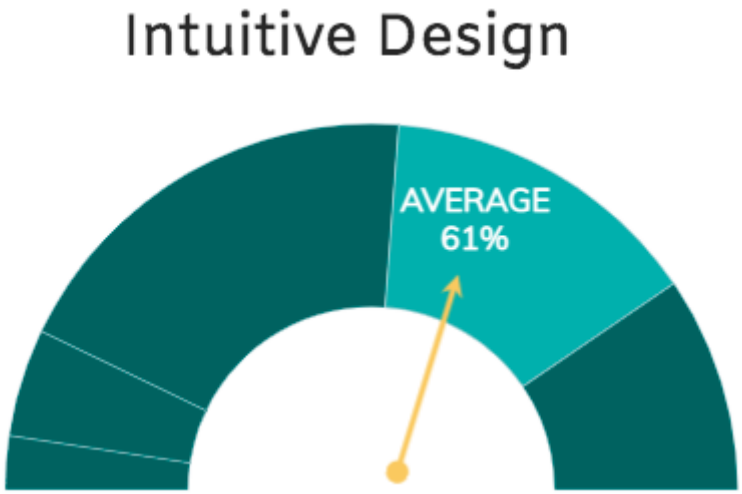
6.4 User Testing

For our user testing, we surveyed a small number of family members and friends on their experience using our application. We gave them brief instructions on how to set up their profile, add a bluetooth device to their list of authenticated devices and observed how they used the application. We made sure not to give instructions that were too detailed as we were very interested to see how easy they found getting to where they wanted to be while navigation through the application. The feedback and criticism we received was extremely helpful, especially from the users without a

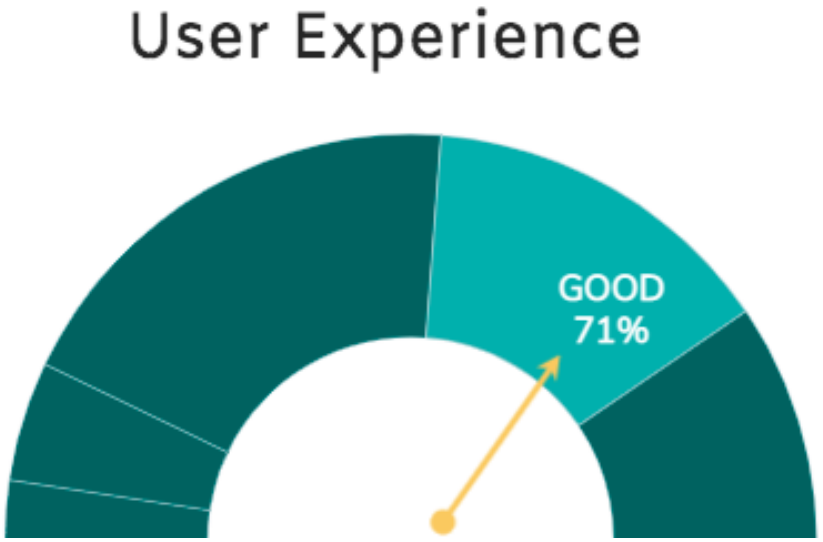
background using computers. Users reported that they were slightly “overloaded” with information as they didn’t know where to look upon entering the site for the first time. As a result of this feedback, we scaled back the number of items on each page and also increased the size of buttons that we felt would aid users in finding what they were looking for. As we had multiple users report similar criticisms, the feedback we received was not great in terms of performance, but we were very happy with the honesty as it allowed us to identify and fix these issues with our application.

6.5 Ad-Hoc Testing

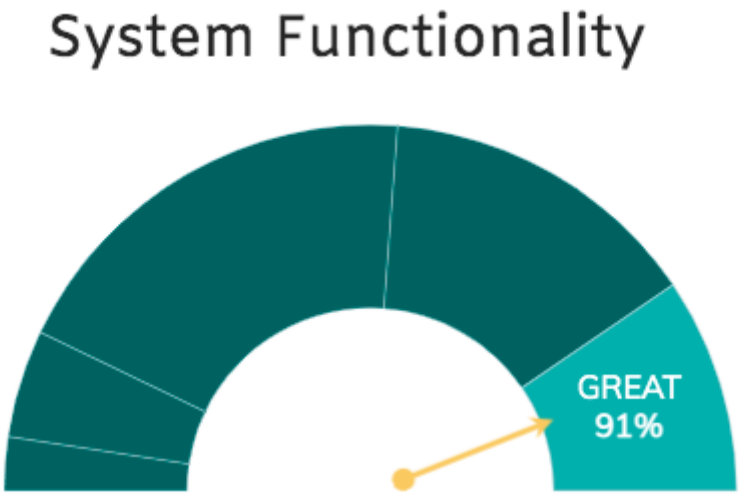
As we developed and tested our application, we focused on attempting to purposely “break” some of the functional components on our website. This type of testing was extremely useful to us as it allowed us to identify areas that we could improve on that would most likely not be identified by an average user, but could definitely still occur.



As a result we did not do too well regarding our intuitive design responses. But overall the feedback regarding the user experience was quite positive, which we believe stems from our great responses to questions regarding the overall system functionality.



We fared exceptionally in this regard, with 91% of the feedback being overwhelmingly positive. We attribute our high satisfaction rating with the system functionality to our testing processes, as we made sure to test every functional component of the application thoroughly in order to achieve such a high level of system functionality.



7. Results

7.1 Future Work

We believe that we could significantly improve the overall quality of our device and application if we do decide to continue to work on this project in the future. This could be achieved in a number of ways, both hardware and software related. By adding a lot more features to our application, this would allow for greater flexibility and user control of the device. As we are under time constraints, and because of the amount of testing required for our project, it was difficult to implement as many features as we would have liked to. On the hardware side of things, we could not only improve the overall performance of our device by using more expensive hardware modules, but could also design a much more space efficient product that would only incorporate what is needed by our application. As we decided to implement collision detection towards the end of our project, we also feel like we could have improved the standard of our implementation significantly. It would have also been beneficial to have had the opportunity to work with more hardware components, like a microphone for example, which can also be extremely useful in collision detection.

7.2 Conclusion

We are grateful to have gotten the opportunity to learn new frameworks and tools throughout the course of this project, and we are amazed by how much we managed to expand on our original idea in our project proposal. As we learned more and more about the project, considerations that we were unaware of were brought to light, which kept us on our toes as we were always thinking about not only how we could design the system to the highest level for it's current specification, but also how it might look in the future in a production environment. We were constantly "putting out fires" throughout this project, which really kept it fresh and made the experience that much more enjoyable in the end. There really is no better way to learn than picking something you aren't familiar with, and going through the building (and breaking) process as a team, learning every step of the way. This project has also given us a chance to reflect on areas we may need to improve upon, and also those in which we have come a long way in during our four years here at DCU.