Functional Specification for: Thieft

Prepared by Jack O'Reilly & Niall Bermingham

17/11/20

Table of Contents

Table of contents

1. Introduction

1.1 Purpose

1.2 Overview

1.3 Business Context

1.4 Glossary

2. General Description

2.1 Product / System Functions

2.2 User Characteristics and Objectives

2.3 Operational Scenarios

2.4 Constraints

3. Functional Requirements

3.1 User Login

3.2 User Profile

3.3 Detect Possible Theft Incident

3.4 Alert Possible Theft Incident

3.5 Detect Possible Device Tampering

3.6 Alert Possible Device Tampering

3.7 On-Board Diagnostics (OBD) Interface

3.8 Security

3.9 Hardware Integration

4. System Architecture

4.1 Figure A - System Architecture Diagram

5. High-Level Design

5.1 Figure B - Context DIagram

5.2 Figure C - Data Flow Diagram

6. Preliminary Schedule

6.1 Overview

6.2 Figure D - Gantt chart

1. Introduction

1.1 Purpose

The purpose of this document is to give a synopsis of our multi-purpose anti-theft device for vehicles, Thieft, and will provide a reference for the designing of the system. The intended audience is the project coordinator, project supervisor and the CA400 demonstration panel. The document will also include detailed information on specified requirements, will identify constraints on the application and will explain how any concerns of stakeholders are met.

1.2 Overview

Our project, Thieft, is a bluetooth-enabled anti-theft device for vehicles that also hosts a graphical user interface for the On-Board Diagnostics(OBD) of the car. The device incorporates a Raspberry Pi, Accelerometer, GPS and SIM card. The Raspberry Pi single-board computer has built-in bluetooth connectivity, which enables us to detect whether or not the owner of the vehicle's phone is in range of our device. The accelerometer will allow us to tell if the vehicle is moving or not. If the vehicle is moving, and the owner's phone is not in range of the device, an SMS alert will then be sent to the owner, which will provide the location of the vehicle provided by the GPS. The device will also host an intuitive and educational graphical user interface of the vehicle’s On-Board Diagnostics (OBD), when connected to the OBD port of the vehicle. The interface will be designed to suit users who have no experience with OBD, and will allow them to examine possible issues with the vehicle’s various systems, and make decisions based on results/output levels.

1.3 Business Context

Between 2015 and 2017, roughly 700,000 motor vehicles were stolen across the EU. 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. To give an example, many relatively new cars have keys that send a short-range signal that tells them to unlock, even if they are in your pocket. While this technology is certainly very useful, unfortunately, it has become common practice for car thieves to 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. As there are many other vehicle related capabilities using a Raspberry Pi, we thought we could make our device even more useful by allowing users to access the On-Board Diagnostics(OBD) of their vehicle, through an intuitive graphical interface. This will allow users to make decisions based on this information, and seek further assistance in the event they are alerted to a possible issue/fault in one of the monitored vehicle systems.

1.4 Glossary

OBD - On-board diagnostics (OBD) is an automotive term referring to a vehicle's self-diagnostic and reporting capability.

GUI - A graphical user interface (GUI) is a type of user interface through which users interact with electronic devices via visual indicator representations.

SMS - SMS stands for Short Message Service, and it's the most common form of text messaging used today.

GPS - The Global Positioning System (GPS) is a navigation system using satellites, a receiver and algorithms to synchronize location, velocity and time data for air, sea and land travel.

HTML - HyperText Markup Language is the standard markup language for displaying documents in the web browser.

CSS - Cascading Style Sheet is a style sheet language for deserving the presentation of a markup language document.

2. General Description

2.1 Product / System Functions

User Functionality

The web application will be accessed through a web browser by the user, who will then login using their unique credentials, and have the ability to access each area of the application from the homepage. Users will be able to update their contact information, change their login and wifi credentials, and view the status of each of the monitored on-board diagnostics of the vehicle via the graphical user interface. Users can also find a tutorial on the web app that will guide them through each of the possible functions they are able to perform, and will also guide them through the initial setup process of the device (connecting to the vehicle, adding their phone number, how to access the OBD etc.).

GUI Functionality

The graphical user interface will be displayed to the user upon connecting to the device via bluetooth. Initially, users will be met with a login page, where they will enter their username and password. These values are then hashed using a secure hashing algorithm (SHA-256) and stored in a file in bit form. Once logged in, users can perform the various functions mentioned above that are part of the tutorial, as well as view the vehicle's on-board diagnostics. The OBD information displayed is real time sensor data, obtained using the Python-OBD library. The OBD interface operates in a request-reply fashion, so in order to retrieve data from the vehicle, you must send commands that query the data that you want. We enable users to perform these commands in an easy to understand way, and also provide them with an overview of each of the monitored OBD that will allow them to identify any issues/faults with any of the tracked vehicle systems, at a glance.

Web-Interface Functionality

The web interface will be primarily developed using Javascript with React.js, in a Node.js environment, with some HTML, and will be styled with CSS. Python and C++ will be used for the various backend components and hardware implementation. The OBD information will be retrieved as JSON data, before being parsed, formatted, styled appropriately and displayed to the user in an easily understandable manner.

2.2 User Characteristics and Objectives

This device is appropriate for anyone who owns a vehicle, once users have access to a phone with bluetooth and wifi connectivity enabled, they can establish a connection to the device. Users require no prior knowledge of vehicles as we aim to provide them with an easy to understand interface, which offers information about the vehicle's various systems in an intuitive and educational way. The only user requirement to use our web application is the ability to read and understand English although certain disabilities/impairments may affect one's ability to set up the device and/or learn about the OBD of the vehicle. Accessing each component of the application should be a seamless and effortless process, to improve usability and increase overall user satisfaction.

2.3 Operational Scenarios

Use Case 1 - Alert For Possible Theft Incident

Once a user has logged in and set up the device, if the user’s phone is not in range of the device and the user’s vehicle starts moving, they will receive an SMS Alert containing details of the incident. The message they receive will provide them with the date, time and location of the vehicle when the incident was detected. Once the user has verified that the vehicle in question has been stolen, they can quickly contact emergency services with this information.

Use Case 2 - Alert For Possible Device Tampering

Once a user has logged in and set up the device, if the device is disconnected from the vehicle’s OBD port, the user will receive an SMS Alert containing details of the possible device tampering incident. This alert will also provide the user with the date, time and location of the vehicle when the incident was detected. The user will be instructed to proceed with caution when verifying the incident, as the vehicle in question may be in the process of being stolen. Once the device tampering alert is verified by the user, they can quickly contact emergency services with the information about the vehicle that we have provided in the alert.

Use Case 3 - View Vehicle On-Board Diagnostics (OBD)

When the user has logged in and connected the device to the OBD port of the vehicle, they have the ability to access the OBD interface on the web app. Here the user can examine each of the monitored OBD of the vehicle, and will also be notified of a possible issue/fault in any of the vehicle’s various systems that are tracked by our device. Users can also view a more detailed section for each of the OBD, as we provide them with more information about each of the monitored OBD in an educational manner.

Use Case 4 - Notification Of OBD Issue/Fault

When the user has logged in and connected the device to the OBD port of the vehicle, the moment they view the OBD interface, the OBD sensor data is streamed directly from the vehicle, before being parsed, formatted, styled appropriately and presented to the user. The user can immediately determine if there is an issue/fault with one or more of the monitored OBD through this interface, as the data is measured against the expected results/output levels upon retrieval. If the data is within the expected range for one of the OBD, the status colour displayed to the user will be green, if not then the colour will be red. This allows the user to instantly recognise that there may be a problem with the OBD, they can then select that OBD for a more detailed explanation as to why an issue/fault has occurred.

2.4 Constraints

Below are the possible constraints that may be placed upon us while designing this device.

Time

As we are extremely busy with many assignments/studying for exams, much of the code for the project will most likely be developed after the first exam period as a result of this. We have placed a large emphasis on planning as a result of this, and will ensure to allow an appropriate amount of time for each part of the project.

Testing

An extensive amount of testing must be carried out during the course of this project. As there are very serious consequences for something going wrong with our device, we will have to adequately test each component we implement, this will also help us to ensure code quality standards are maintained at all times.

Security

It will be extremely important for us to consider how we will secure both the device and web application from all forms of tampering. This includes the sensitive information we receive from the user (username and password) as well as securing the information from the connected device against possible attacks, like MAC address spoofing.

3. Functional Requirements

3.1 User Login

Description

The application must allow for a user to login using their username and password. Default login values will be located on the device, which the user will be instructed to change and remember upon their first login.

Criticality

This login is essential as without it, an attacker could easily change the contact number for the device before disconnecting it from the OBD port of the vehicle. The user would then not receive any SMS alert, allowing the attacker the ability to drive off in the user's vehicle without being detected.

Technical Issues

How we implement the different hashing algorithms and store the user's data is of utmost importance. We must ensure that this data is not freely available to other users, and not susceptible to attackers using brute force, and other methods of password/key decryption.

Dependencies with Other Requirements

This requirement will be dependent on functional requirement 3.8 as it is extremely important that the user's credentials are protected and stored effectively to ensure data integrity.

3.2 User Profile

Description

The application will have a profile for each user, and will remember the user's device, username, password and contact number. Users will also have the option to include more details about their vehicle, such as the make, model and registration number. This information will be included in the SMS alert should a possible theft/tampering incident occur.

Criticality

This requirement is critical to our system as the user profile contains all of the necessary information about the user and their vehicle, making it a very important part of our application.

Technical Issues

A large focus will need to be placed on the protecting, storing and also the presenting of this data to the user. The interface for the user profile should allow for a simple and easy way for users to update and manage the information provided to us.

Dependencies with Other Requirements

This requirement will also be dependent on functional requirement 3.8 as we must ensure that the data provided by the user on their profile is efficiently managed and stored safely. It is also dependent on functional requirement 3.1 as the user will need to be able to login before updating their user profile.

3.3 Detect Possible Theft Incident

Description

The application will be able to detect incidents of possible theft, using the various hardware components outlined above. When the device has been set up and is connected to the OBD port of the vehicle, the bluetooth module of the RaspberryPi will detect if the user's phone is within range of the device. The accelerometer allows us to tell if the vehicle is moving or not, if the owner's phone is not in range of the vehicle and the vehicle is moving, this will be declared as a detection of a possible theft incident.

Criticality

This is extremely critical to our application, if this requirement is not developed correctly, then half of our entire application will serve no purpose. This requirement is arguably the most critical, we must devote our time to developing this feature accordingly, and take ample time to consider all possible issues we may face in the development and implementation process.

Technical Issues

There are many technical issues that may arise when working on this requirement, as we will be largely depending on the ability to send and receive data to and from each of the hardware components that are part of our device. Should the GPS fail, for example, we would no longer be able to tell the location of the user's vehicle.

Dependencies with Other Requirements

As mentioned above, this requirement is largely dependent on functional requirement 3.9 as the hardware components will be responsible for tracking much of the data we plan to collect.

3.4 Alert Possible Theft Incident

Description

The application will also be able to alert users to incidents of possible theft after detection, using the data collected from various hardware components outlined above, and sending the information to the user using the SIM card within the device. The user can then verify if the vehicle in question has been stolen, and contact emergency services with the information provided to them if it has.

Criticality

This is also extremely critical to our application, as if this requirement were to fail, the user would be left without receiving an alert to their vehicle possibly being stolen. This is a very important part of our application, and this requirement will have to undergo a great deal of testing as a result.

Technical Issues

There are also many technical issues that may arise when working on this requirement, as any failure in the messaging/alert system would result in the user's vehicle possibly being stolen with no warning to them. If the SIM card was to stop working, for example, there would be no SMS alert sent to the user.

Dependencies with Other Requirements

This requirement is dependent on functional requirement 3.3 - as the possible theft incident will first have to be detected, and 3.9 - as the hardware components will be responsible for retrieving the information needed to alert the user.

3.5 Detect Possible Device Tampering

Description

The application will be able to detect incidents of possible device tampering, when an incorrect password is entered multiple times, or when the device is disconnected from the OBD port of the vehicle . When the device has been set up and is connected to the OBD port of the vehicle, if it is then removed from the port it will trigger an incident of possible tampering to be detected.

Criticality

This is an extremely important requirement of our application, as without this requirement there is nothing stopping a thief from simply disconnecting the device, removing it entirely and driving off with the stolen vehicle.

Technical Issues

It is possible that users could accidentally set off tampering alerts, we must also ensure that a viable forget password system is put in place.

Dependencies with Other Requirements

This requirement is largely dependent on functional requirement 3.1 as the login page will have to account for a situation where a user has forgotten their password.

3.6 Alert Possible Device Tampering

Description

The application will also be able to alert users to incidents of possible tampering when detected, and will send the information of the incident to the user. The user can then verify if the device is being/has been tampered with, and contact emergency services with the information provided to them if it is/has.

Criticality

This requirement is also of great importance to our application, as if this requirement were to fail, the user would be left without receiving an alert to their vehicle possibly being stolen.

Technical Issues

Any failure in the messaging/alert system would result in the user's vehicle possibly being stolen with no warning to them. If the SIM card was tampered with and was to stop working, for example, there would be no SMS alert sent to the user.

Dependencies with Other Requirements

This requirement is dependent on functional requirement 3.5 - as the possible tampering incident will first have to be detected, and 3.9 - as the hardware components will be responsible for retrieving the information needed to alert the user.

3.7 On-Board Diagnostics (OBD) Interface

Description

The application will also host a graphical user interface for the OBD of the vehicle, which will provide users with an overview of the monitored OBD, and will also allow the user to examine the results/output levels for each of the individual OBD.

Criticality

This requirement is not of the same level of importance as some of the other requirements, but certainly a useful feature of the web application nonetheless.

Technical Issues

If a query to the OBD of the vehicle is made incorrectly, or no data can be received, then we will be unable to display the results/output levels to the user.

Dependencies with Other Requirements

This requirement will be dependent on functional requirement 3.1 - as users must be logged in while the device is connected to the OBD port if they wish to view information about the OBD of the vehicle.

3.8 Security

Description

There are many considerations for us to take into account when ensuring a secure application for our users. How we retrieve and store data is very important, and we plan to use the SHA-256 hashing algorithm to secure passwords and other sensitive information.

Criticality

This security of our application is one of the most critically important requirements, thieves have become increasingly sophisticated as vehicle technology has advanced rapidly over the past decade. We must do all we can to prevent brute force and other attacks on our system, and protect our user's information to the best of our ability.

Technical Issues

How we implement the different hashing algorithms and store the user's data is of utmost importance. We must ensure that this data is not freely available to other users, and not susceptible to attackers using brute force, and other methods of password/key decryption.

Dependencies with Other Requirements

This requirement will be dependent on functional requirement 3.1 as it is important that the user's credentials are protected and stored effectively when a user logs into our application.

3.9 Hardware Integration

Description

Our device will incorporate many different hardware components, these include: an Accelerometer, a Raspberry Pi, a GPS and a SIM card. The accelerometer will be used to detect if the vehicle is moving. The Raspberry Pi will host the source code of the application, as well as provide the bluetooth and wifi connectivity we need for different features mentioned above. The GPS will provide the location of the vehicle and the SIM card will be used to alert users via SMS.

Criticality

Each of the various hardware components we are integrating are very important. If one of these components fails, a large portion of source code will no longer work. For example, if the Accelerometer should fail, we will no longer be able to detect if the vehicle is moving or not.

Technical Issues

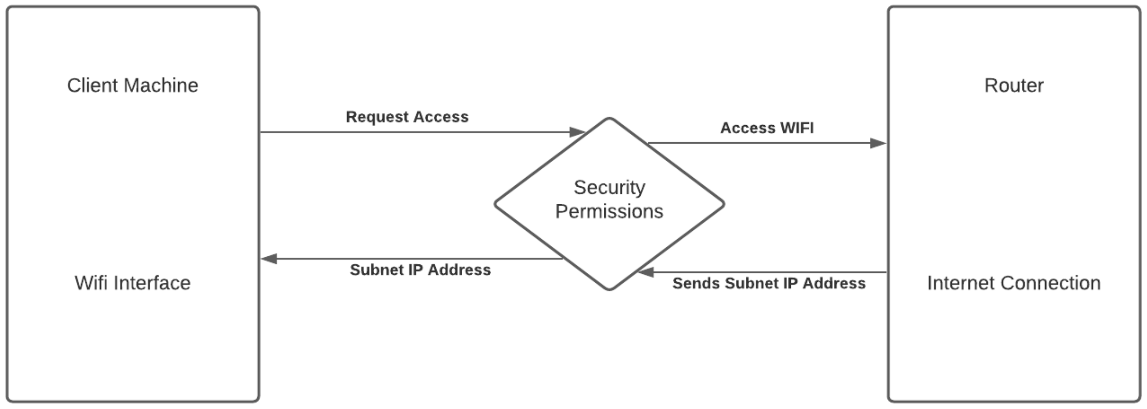
We expect to experience some issues from time to time with the hardware we are using. During our initial testing, we had problems with the GPS, as it took a long time to initialise and send data back to our system.

Dependencies with Other Requirements

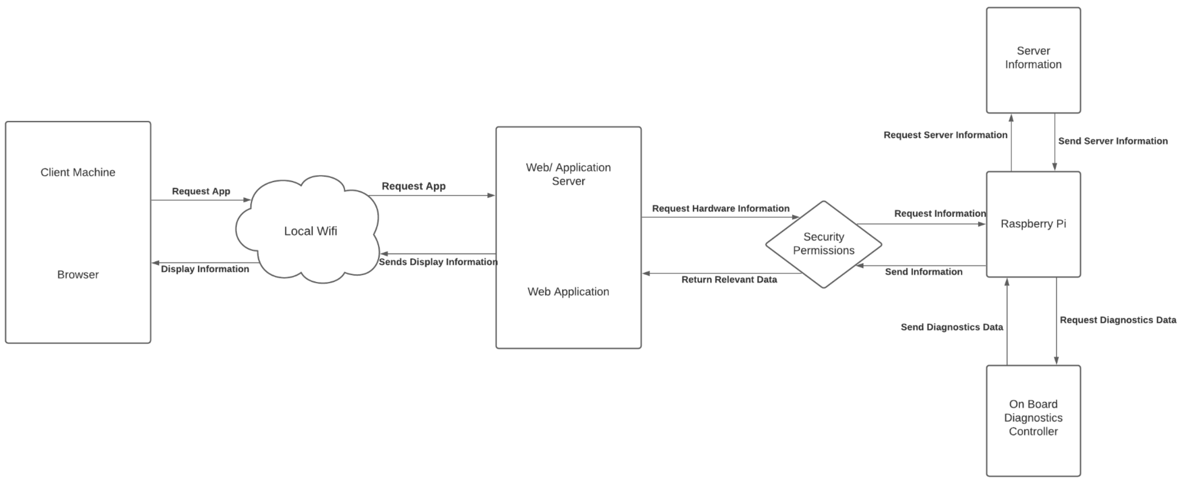
Much of the web application is entirely dependent on these hardware components. The alert system is entirely dependent on the SIM card to be operational so it can alert the user via SMS. The detection systems are dependent on the accelerometer and bluetooth connectivity of the Raspberry Pi.

4. System Architecture

4.1 System Architecture Diagram - Router Connection



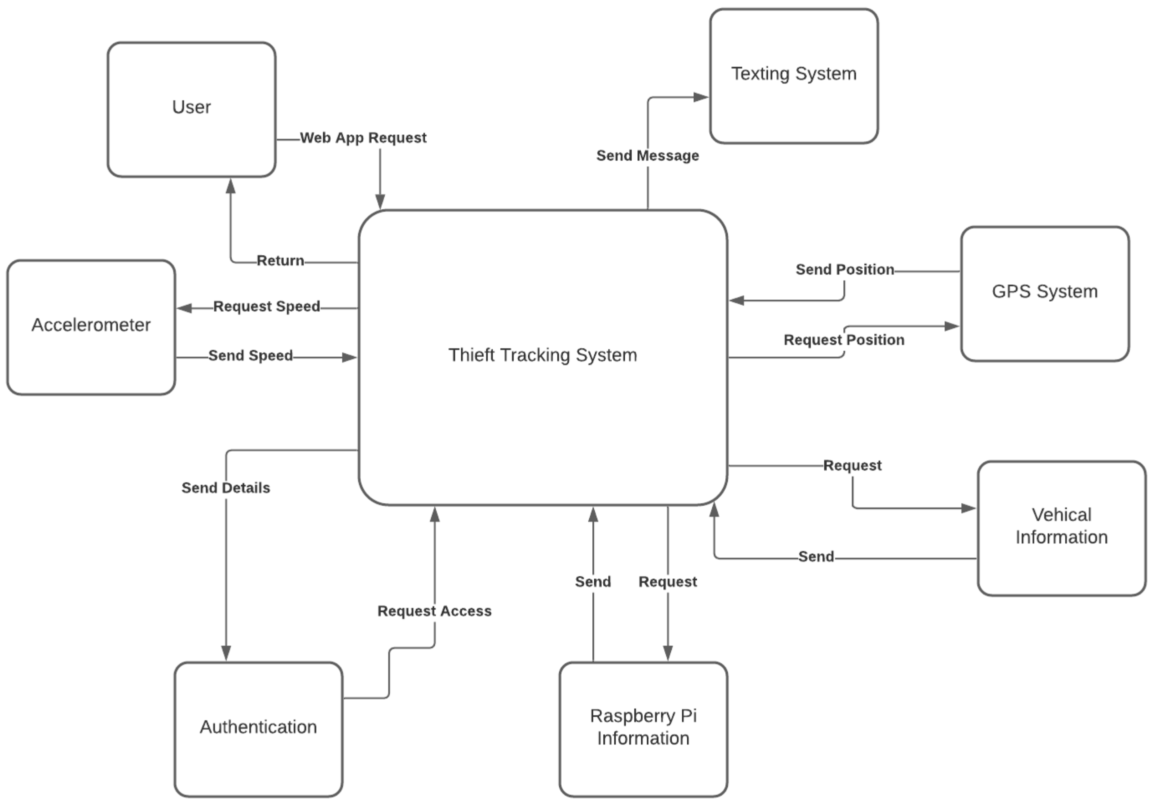
4.2 System Architecture Diagram - Thieft System



5. High-Level Design

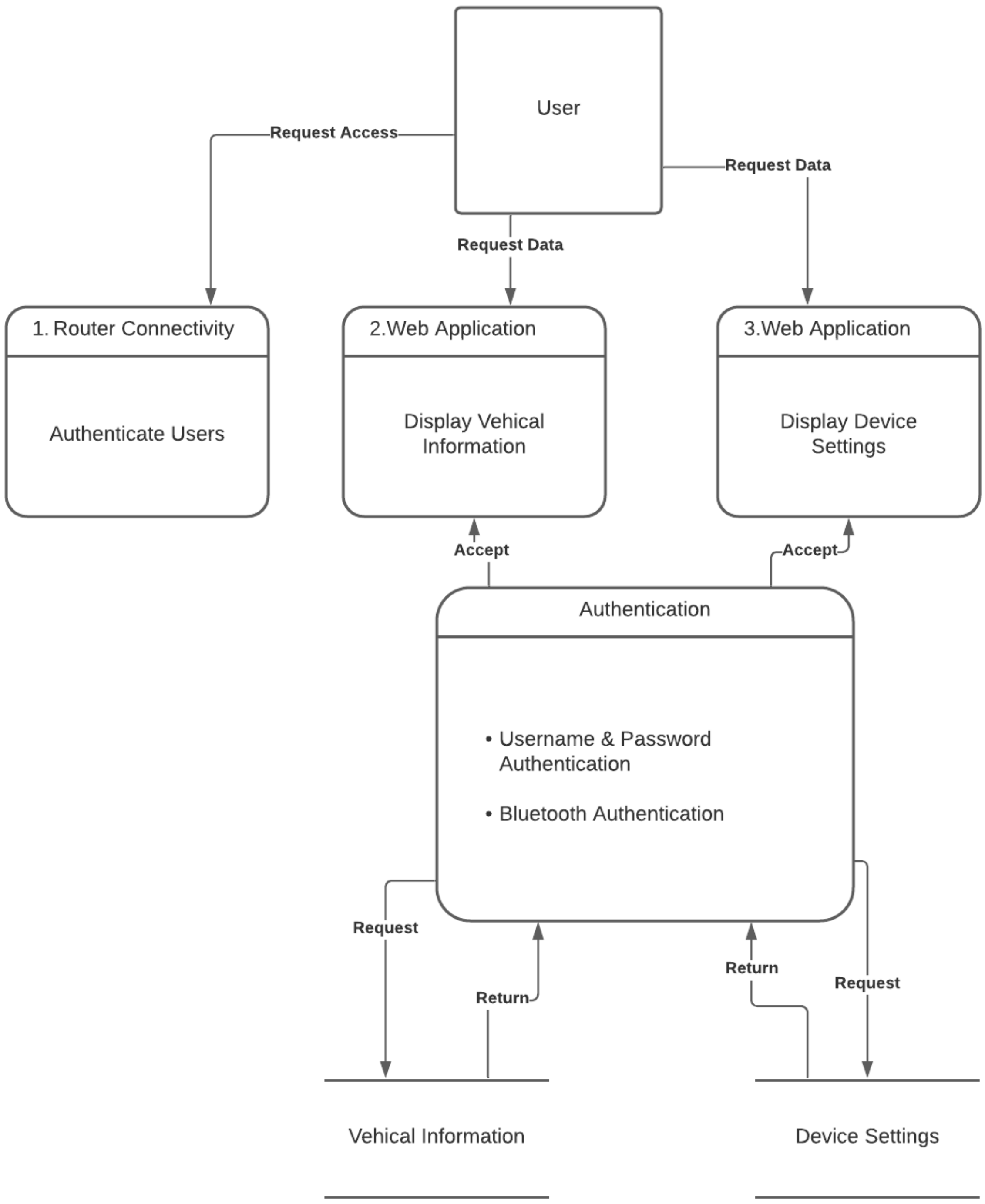
5.1 Context Diagram

The context diagram is a basic overarching look at the system architecture. There is one type of user that is required to login using an username and password. The system will check the login details and check the users device to be authorised via bluetooth. The User will be able to search through the vehicle's information, GPS location and update/change their user details e.g user login credentials and the authorised devices that can connect to the system.



5.2 Data Flow Diagram

The data flow diagram shows a more detailed view of interactions between the user and the device. Authorised users can connect to the device via WIFI and then load the web app in their browser. Users can add devices and change their authentication details as well as view vehicle status and the system's settings. This information will be locked behind a security authentication to prevent unauthorized users from connecting.



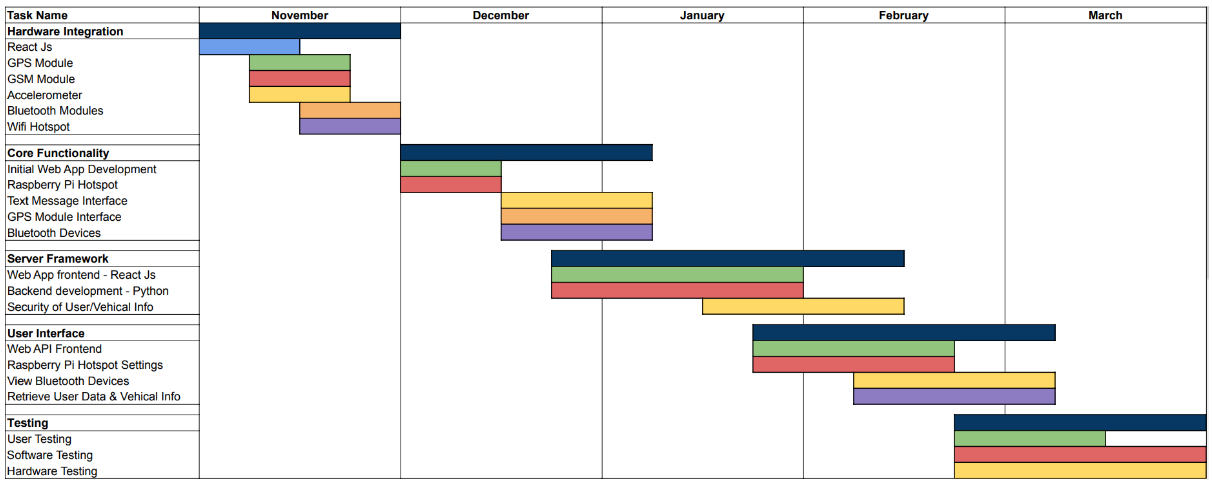
6. Preliminary Schedule

6.1 Overview

The image below is of our Gantt chart which was created using Google Sheets. The chart shows a detailed timeline of how our proposal has come along so far and outlines a clear plan of what we hope to accomplish in the future. On the left, you can see all of the tasks we are taking into account in the project so far. We will look in detail at the tasks involved in the implementation in 6.2.

6.2 Detailed View

In the Gantt chart below, you can see an in-depth breakdown of how we plan to allocate time for our implementation. We will start with about one month on setting up hardware integration in our project. This involves wiring up hardware and retrieval of information from those hardware modules. Following that is creating the core functionality of our application which we would like to spend 5 weeks on development. This would include implementation of some of the basic features we would like to include like setting up our Raspberry PI for wifi connectivity and setting up a React web application to be used in our project. Next we would like to spend 7 weeks on the Server Framework. This will be the bulk of the project as it includes taking all of the information we retrieve into a usable format for our web application. Along with this we will be implementing security measures to prevent unauthorised access to the server as this will be a large section we decided we should designate the largest portion of time to this section. Our next section will be building the User interface for the project. This will be how the user interacts with our product. We wish to create a friendly and easy to use interface so it will be accessible to people without much technical knowledge and be easy to set up if someone were to install this anti-theft device into their car. Finally we hope to spend the remaining time on testing. This is an anti theft device so we want to make sure it is fully functional and that our security measures are strong enough to stop any brute force attack on the system. We will also be doing user testing, this would entail asking testers to set up this device on their own using the tutorial we provide, and giving them a list of actions we would like them to perform. The feedback we receive we hope to have implemented with the remaining time left for the project.


Task Name
November
December
January
February
March
Hardware Integration
React Js
GPS Module
GSM Module
Accelerometer
Bluetooth Modules
Wifi Hotspot
Core Functionality
Initial Web App Development
Raspberry Pi Hotspot
Text Message Interface
GPS Module Interface
Bluetooth Devices
Server Framework
Web App frontend - React Js
Backend development - Python
Security of User/Vehical Info
User Interface
Web API Frontend
Raspberry Pi Hotspot Settings
View Bluetooth Devices
Retrieve User Data & Vehical Info
Testing
User Testing
Software Testing
Hardware Testing