Perfect Parking

An AI Application to Assist Drivers Finding Parking in Busy Cities

Rhys Quilter

K00241356

A Final Year Project submitted as a requirement of the Technological University of Shannon for the degree of Bachelor of Science (Honors) in Software Development.

Supervised by:

John Jennings

Acknowledgments

I would like to thank my supervisor John Jennings for helping me to complete my research. In addition, I would like to thank my parents Sean and Beverley, and my girlfriend Laura for their support during my time at TUS.

Abstract

*This project is a Parking application for academic purpose. The aim of Perfect Parking is to create a parking system that will replace outdated systems and to help stop the widespread problem that is parking in our cities. The applications that are used in Limerick City are simply not good enough. So, the goal in this project is to improve the effectiveness of finding parking spaces and to also relieve the stress of the users looking for parking by implementing new and innovative features. I will do this by using the Django framework structure and by implementing methods such APIs that will show the user exactly where the parking is and by providing locations for them to follow straight to the location. I will also use a parking monitor powered by OpenCV to detect if a parking spot has been filled or made empty. By doing this I feel like it will also help with traffic congestion in the city as people won’t need to keep driving around the block to find a convenient parking space.*

# Table of Contents

[Acknowledgments ii](#_Toc133322527)

[Abstract iii](#_Toc133322528)

[Table of Figure viii](#_Toc133322529)

[Chapter 1 Introduction 9](#_Toc133322530)

[1.1 The academic objectives 9](#_Toc133322531)

[1.2 Problem Statement 9](#_Toc133322532)

[1.3 Perfect Parking: a solution 10](#_Toc133322533)

[1.4 Objectives 10](#_Toc133322534)

[1.5 The Scope of the solution 10](#_Toc133322535)

[1.6 Report Structure 11](#_Toc133322536)

[Chapter 2 Literature Review 12](#_Toc133322537)

[2.1 Big Data and Realtime Data 12](#_Toc133322538)

[2.1.1 What is big data? 12](#_Toc133322539)

[2.1.2 Why is big data important? 12](#_Toc133322540)

[2.1.3 What is Real Time data 13](#_Toc133322541)

[2.1.4 What is the importance of Real Time data? 13](#_Toc133322542)

[2.2 Problems with gathering data. 14](#_Toc133322543)

[2.1 Object Recognition and AI 14](#_Toc133322544)

[2.1.0 Machine Learning 14](#_Toc133322545)

[2.1.1 Computer Vision 15](#_Toc133322546)

[2.1.2 Object Detection 15](#_Toc133322547)

[2.1.3 Cascading classifiers 15](#_Toc133322548)

[2.1.4 Haar-like feature 16](#_Toc133322549)

[2.2 Conclusion: The Need for a Software Solution 17](#_Toc133322550)

[Chapter 3 Project Management 19](#_Toc133322551)

[3.1 Weekly Meetings 19](#_Toc133322552)

[3.2 Source code management (SCM) 19](#_Toc133322553)

[3.3 Code Style Guide 19](#_Toc133322554)

[3.4 Collaboration Tools 19](#_Toc133322555)

[3.4.1 GitHub 19](#_Toc133322556)

[3.4.2 Microsoft Office Online 19](#_Toc133322557)

[Chapter 4 Analysis and Design 20](#_Toc133322558)

[4.1 Unique Selling Point 20](#_Toc133322559)

[4.2 The Application 20](#_Toc133322560)

[4.3 Users Use Case Diagram 21](#_Toc133322561)

[4.4 System Actors 21](#_Toc133322562)

[4.5 Use Case Descriptions 21](#_Toc133322563)

[4.5.1 Use Case: Find Parking 21](#_Toc133322564)

[4.5.2 Use Case: Register User 22](#_Toc133322565)

[4.5.3 Use Case: Update Parking Lot Status 23](#_Toc133322566)

[4.5.4 Use Case: User changes password. 24](#_Toc133322567)

[4.6 Identifying the free/busy car parking spaces 25](#_Toc133322568)

[4.7 Machine Learning / Artificial Intelligence (AI) 25](#_Toc133322569)

[4.8 Sensors 25](#_Toc133322570)

[4.8.1 Development limitations 26](#_Toc133322571)

[4.9 Machine Learning Algorithms 26](#_Toc133322572)

[4.10 Database design 26](#_Toc133322573)

[4.11 User Parking Sequence diagram 27](#_Toc133322574)

[4.1 Introduction and focus 28](#_Toc133322575)

[4.2 Academic Aims 28](#_Toc133322576)

[4.2.1 Academic Requirements 28](#_Toc133322577)

[4.3 Functional Requirements 28](#_Toc133322578)

[4.4 Non-Functional Requirements 28](#_Toc133322579)

[4.5 Statistics 28](#_Toc133322580)

[Chapter 5 Implementation 29](#_Toc133322581)

[5.1 Object Recognition in Images 29](#_Toc133322582)

[5.2 How Object Recognition and AI Is Used in Perfect Parking 29](#_Toc133322583)

[5.3 Client and Server Architecture with Rest framework 30](#_Toc133322584)

[5.3.1 The Monitor 30](#_Toc133322585)

[5.3.2 Server 32](#_Toc133322586)

[5.3.3 How They Work Together 33](#_Toc133322587)

[5.3.4 Source Control and versioning 33](#_Toc133322588)

[5.4 Development Environment 33](#_Toc133322589)

[5.5 Tools Used 33](#_Toc133322590)

[5.6 Django Rest API 34](#_Toc133322591)

[5.6.1 Perfect Parking with Django 34](#_Toc133322592)

[5.7 Anaconda 35](#_Toc133322593)

[Chapter 6 Conclusion and Recommendations 36](#_Toc133322594)

[6.1 Conclusion 36](#_Toc133322595)

[6.2 Recommendations 36](#_Toc133322596)

[References 37](#_Toc133322597)

[Glossary 38](#_Toc133322598)

[Appendix A Reflections 39](#_Toc133322599)

[A.1 Report Structure 39](#_Toc133322600)

[Appendix B Project Management 40](#_Toc133322601)

[B.1 Report Structure 40](#_Toc133322602)

[B.2 Code Style Guide 40](#_Toc133322603)

[B.2.1 Naming conventions 40](#_Toc133322604)

[B.2.2 Avoid magic constant numbers. 40](#_Toc133322605)

[B.2.3 Variable naming 40](#_Toc133322606)

[B.2.4 Methods 40](#_Toc133322607)

[B.2.5 Imports 40](#_Toc133322608)

[B.2.6 Comments 40](#_Toc133322609)

[B.2.7 Documentation 40](#_Toc133322610)

[B.2.8 Classes 40](#_Toc133322611)

[B.2.9 Spacing, Indentation 40](#_Toc133322612)

[B.2.10 Literals 40](#_Toc133322613)

[Appendix C Development Environment 41](#_Toc133322614)

Table of Figure

[Figure 1TUS Logo 16](#_Toc132726956)

[Figure 2- User Use Case Diagram 17](#_Toc132726957)

[Figure 3 - Database Design 23](#_Toc132726958)

[Figure 4- User Parking Sequence Diagram 24](#_Toc132726959)

# Introduction

Due to an increase in in the number of cars being used in limerick city and other cities in Ireland, finding a solution to car parks has now become vital. The old-fashioned way of parking was that everyone would just leave their cars parked in the streets until they were needed again, this however caused major traffic congestions in towns and cities. Shop owners also got hugely impacted as there wouldn’t be enough room for staff to park no mind the customers looking to go into their shops which was damaging for their business, it is undeniable that car parks are a very important factor in society, by having parking spaces it reduces illegal parking which would have increased congestions on the road and increasing travel time.

Parking back in the 1980s – 1990s wasn’t such a big issue as there wasn’t very many cars on the road as people couldn’t afford to have a car unless they were wealthy, nowadays however it is very hard to find available parking spaces in places such as cities, and of course universities especially during rush hour. Since limerick City is a big city for students to come and study in with there being over 16,000 students attending university of limerick and just under 2000 students in TUS and Mary I, this brings so much more motor vehicles into limerick city which is a city already struggling with car parking. (University of Limerick, 2022)

The goal is to try and reduce this issue by building a new innovative parking app and to try to help motorists stop stressing about this parking crisis.

## The academic objectives

The academic objectives of this project are to study and gain experience working with AI and object detection in images.

The chosen problem used for this study is to help to reduce the traffic congestion in cities such as Limerick.

## Problem Statement

**Ineffective ways of finding an available parking space which is a waste of time, very fuel consuming and causes traffic jams.**

when road users are looking to find an available parking space they end up wasting time and using a lot of fuel from them driving around the car park or the block multiple times hopping to find a space, on average people spend 17 hours per year driving around looking for parking spaces (Quellmalz, 2021). By developing Perfect Parking, It is hoped to make the parking process in college campus and in the city seamless and stress free, by doing this I’m hoping to eliminate the time and fuel waste road users encounter while looking for parking.

**Lack of visual parking space availability.**

When road users are driving the visuals of the eye are limited, the vision is blocked by many obstacles such as the cars frame causing blind spots, other cars, trees and much more. It can be understood that the road users can find it difficult to spot an available parking space if it’s in the distance or behind other cars. This has turned into an important problem with road users and as a result they waist time and fuel trying to spot an available space.

**Lack of knowledge of towns or cities.**

Limerick is a city where people migrate to for education or tourism, and with this brings more road users. When drivers first come to Limerick City, they must learn the road routes and with this where the parking is. A lot of road users that drive in a new place start to panic and get anxious when they try and find parking. This can cause them to be in the wrong lanes and cause traffic congestion. But with the Perfect Parking app it will allow users to plan their route to the parking of their choice and follow directions on the phone to the car park. By doing this that it will keep new road users in the city calm so they can enjoy their holiday or for students teach them the road routes and the best places to park.

## Perfect Parking: a solution

## Objectives

## The Scope of the solution

No city or college live feeds

Prerecorded video demonstration car (object) detection and a sever demonstration end user usage and product viability.

## Report Structure

The following is the report structure for the Perfect Parking Thesis. The report has four chapets, lit review, analaysisi, design and colulusions gratitude to those who contributed to the project. The report also includes an Abstract, which provides a brief overview of the project's purpose, scope, methods, and findings. A table of contents and a table of figures are generated automatically, providing a quick and easy way for readers to navigate through the report.

# Literature Review

## Big Data and Realtime Data

A large amount of live data will be required to provide a comprehensive parking software application solution.

### What is big data?

Big data is a combination of structured, semi structured, and unstructured data that is collected by organizations, this data can be mined for information to be used in many projects such as machine learning, predictive modelling, and other analytics applications. Big data is often characterized by the three Vs.

* The large **Volume** of data in many environments.
* The wide **Variety** of data types frequently stored in big data systems.
* The **velocity** at which much of the data is generated, collected, and processed.

These characteristics were first identified in 2001 by Doug Laney, they were then further popularized in 2005 by an analyst at a consulting firm called Meta Group.

Big data doesn’t equate to any specific amount of data, big data deployments often involve terabytes, petabytes or even in some cases exabytes of data that is created and collected over time. (Botelho, n.d.).

### Why is big data important?

Big data importance lies in the fact of how a company utilizes the gathered data. Every company uses its gathered data in its own way, the more a company can gather its data the more the company can grow.

Big data provides valuable insights into customers that companies can use to refine their marketing, advertising, and promotions by doing this they can increase customer engagement and conversion rates. (Anon., n.d.)

Big data is huge in the medical industry, medical researchers can identify disease signs and risk factors, this can help the doctors diagnose illnesses and medical conditions in patients. A combination of data from electronic health records and the web can give healthcare organizations and government agencies up to date information on infectious diseases threats or outbreaks, we have seen this in the past with the pandemic and how the HSE in Ireland were able to monitor the amount of covid – 19 infections per county, and how they were able to create the Covid App with this data. (Botelho, n.d.)

### What is Real Time data

Real time data is data that is available as soon as its created and acquired. Rather than being stored, data is pushed to users as soon as its collected and is immediately available without and delay, this is crucial for supporting live, in the moment decision making. This real time data is a big part of our everyday lives, it powers everything from bank transactions and GPS this was also seen in the many Covid-19 maps that emerged during the pandemic. (Splunk, 2021)

We see a lot more of real time data then we think, Google collects endless amounts of real time data and the way they do it is very smart, they use a device that 6.6 billion people in the world have and that being smart phones. (Turner, 2023) if people have smart phones, then nearly everyone has the google map application and GPS in their phones. When people sign into google on their phone Google starts creating real time data through the GPS and other apps, for example, when your using Google Maps on your phone it shows loads of data such as the estimated time of arrival to your destination and also if there is any traffic on your route, Google knows this by using real time data from other people that are taking that route and that might be stuck in traffic and this is all taken from the GPS location on smart phones. (Ashish, 2022)

### What is the importance of Real Time data?

Real-Time data is a necessity to stay relevant for today’s business and it needs to be delivered by sophisticated electronic communications tools such as digital signage and data dashboards, to remain appealing to today’s tech savvy workforce from call centres to retailers. (Barnett, 2017)

Real time data is important in many parking applications, these applications use real-time data to show users if there is parking spaces in the carpark which they have selected, this data can be gathered by the carpark having a barrier that counts the amount of cars that go in or the amount of cars that exit, some carparks also have sensors on each of the parking spaces this allows users to see what actual spaces are available, this is the most ideal as it allows people that need disabled parking to see if that type of parking space is available.

## Problems with gathering data.

Generic paragraph of the problems of collecting big bag, non-domain specific

## Object Recognition and AI

Object recognition refers to the process of teaching a computer how to identify and classify objects within digital images or videos. It's like teaching a child to recognize different objects such as cars, chairs, or animals. Artificial intelligence, or AI, is the field of computer science that deals with creating machines that can perform tasks that typically require human intelligence, such as learning, reasoning, and problem-solving. AI techniques like deep learning, which is a subset of machine learning, are often used in object recognition systems to train algorithms to recognize and classify objects. This technology has a wide range of applications, from self-driving cars to medical diagnosis to robotics. (Tech Target, n.d.)

### Machine Learning

A subset of artificial intelligence called machine learning involves training algorithms to recognize patterns and correlations in data.

### Computer Vision

The powerful library OpenCV provides a wide range of features for computer vision applications. It is widely used across many different industries, including robotics, driverless cars, medical imaging, and more.

To extract useful information or features, OpenCV processes visual data, such as photos or videos. Image filtering, feature detection, object recognition, and tracking are just a few of the techniques that OpenCV offers to process and analyze visual data. (Boesch, 2023)

### Object Detection

Object detection is a technique in computer vision that involves detecting objects of interest within an image or video stream. Finding the object(s) within a picture and categorizing them into various categories are the goals of object detection. Since object detection requires locating and recognizing multiple objects inside a picture, it is a more advanced technique than object recognition. (Patel, 2020)

### Cascading classifiers

One kind of machine learning technique used in computer vision for object detection is called a cascading classifier. In their groundbreaking study "Rapid Object Detection using a Boosted Cascade of Simple Features" published in 2001, Viola and Jones introduced them for the first time. The approach is based on the concept of "cascading" the solution of a complex detection problem into several smaller, easier sub-problems. (Michael Jones, 2001)

The basic idea behind cascading classifiers is to use a series of classifiers, each with increasing complexity, to detect objects of interest. Utilizing a series of classifiers, each with a higher level of complexity, to find things of interest is the main notion underlying cascading classifiers. The input image is classified as either containing the object of interest or not by each classifier in the cascade using a collection of features. Each classifier's attributes are chosen based on their capacity to distinguish between positive and negative samples. (Lee, 2022)

One of the main advantages of cascading classifiers is their ability to achieve high detection rates with low false positive rates. This is accomplished by employing a number of classifiers, each of which is trained to quickly reject negative samples. As a result, there are fewer false positives because the algorithm can swiftly reject pictures that don't include the object of interest. (Bąk, 2023)

When used in object detection tasks like face detection, cascading classifiers have been shown to be highly accurate and effective. (Bąk, 2023) Several pre-trained cascading classifiers, including the well-known Haar cascades for face detection, are available in OpenCV for object detection.

### Haar-like feature

An image feature type used in computer vision for object detection is called a Haar-like feature. They have the name of the Haar wavelet, which Alfred Haar, a Hungarian mathematician, initially proposed in 1909. (Seal, n.d.) The mathematical function known as the Haar wavelet can be used to break down a signal or image into a collection of wavelet coefficients.

By comparing the average pixel values in adjacent rectangular regions of an image, Haar-like features can be extracted from the Haar wavelet. The difference between the sum of pixel intensities in a rectangular region with a light colour and the sum of pixel intensities in a rectangle region with a dark colour is the precise definition of Haar-like features. (Arunachalam, 2014)

These rectangular areas can be positioned anywhere in the image and come in a variety of sizes and shapes. It is feasible to gather details about the texture and structure of a picture at various levels of granularity by computing Haar-like features at various scales and positions in the image.

The Viola-Jones object detection technique, a well-liked algorithm for face detection in photos, makes use of Haar-like features. In this approach, a classifier is trained to differentiate between positive instances (pictures containing the item of interest, such as faces) and negative examples (images devoid of the object of interest), using a set of Haar-like characteristics computed for each sub-region of an input image. (Tyagi, 2021)

One of the advantages of using Haar-like features for object detection is their computational efficiency. They are suitable for real-time applications like video surveillance since they are rapid and effective to compute utilising integral images. (Bąk, 2023)

## Conclusion: The Need for a Software Solution

In this project, The parking application is being created for educational purposes in an effort to solve the widespread parking issue in our cities. Time is lost, gasoline is consumed, and traffic is backed up due to Limerick City's old and inefficient parking systems. To increase the efficiency of identifying parking spaces and reduce the stress experienced by users searching for parking spaces, a new and creative software solution is required.

Perfect Parking aims to develop a fluid and stress-free parking experience for road users by utilizing the Django framework structure and putting into practice techniques like APIs that display users exactly where parking is available and provide directions straight to the area. Additionally, a parking monitor powered by OpenCV will be used to detect if a parking spot has been filled or made empty, further improving the effectiveness of the system.

The present approaches for locating parking spaces are inefficient and wasteful, requiring a large amount of time and fuel. People spend an average of 17 hours a year searching for parking spaces, according to (McCoy, n.d.) and Irish people waste four days a year. (Sawer, 2017) I intend to end this time and energy waste by creating Perfect Parking, which will be advantageous to both users and the environment.

The lack of obvious availability of parking spaces is another issue with the present parking schemes. Due to the limited vision created by numerous obstructions like the car's frame, trees, or other vehicles, drivers frequently struggle to find an open parking place. Users now spend much more time and fuel because of this issue.

Furthermore, for new road users, finding parking spaces in a new place can be a challenging task. In Limerick City, students and tourists come from different regions to study or visit, and they must learn the road routes and where the parking is. This lack of knowledge can cause them to be in the wrong lanes and create traffic congestion. However, by providing users with the option to plan their route to the parking of their choice and follow directions on their phones to the car park, the application can help users navigate the city's roads and reduce traffic congestion.

To conclude, there is a pressing need for a software solution to address the problems associated with parking in our cities. Perfect Parking aims to provide a solution that is innovative, effective, and user-friendly, with the potential to reduce time and fuel consumption, improve traffic flow, and create a stress-free parking experience for all road users.

# Project Management

under the headings of (i) sub-topic 1 (cf. 1.1.0), and (ii) sub-topic 2 (cf. 1.1.1)

## Weekly Meetings

This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.

## Source code management (SCM)

## Code Style Guide

## Collaboration Tools

### GitHub

### Microsoft Office Online

This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.

# Analysis and Design

## Unique Selling Point

Parking is an issue that contributes to traffic congestion, especially in cities. Cars driving around and around a city for parking adds to the traffic. Cars hovering for parking spots or cars double parked can cause traffic to stop. The purpose of perfect parking is to try and solve the common problem of traffic congestion and scarcity of parking in a city such as Limerick. Perfect parking aims to ease the stress and anxiety that road users face searching for parking by providing live data about availability, pricing, stay-hours, zones, and disability status. Additional benefits include reducing traffic congestion, fuel savings, time and reducing stress.

## The Application

Perfect Parking is a web application that will allow users to find parking in a city. The application will allow users to search for parking near a specific location, and will show the user data the nearest parking to their location.

## Users Use Case Diagram



Figure 2- User Use Case Diagram

## System Actors

* Administrator: The administrator is responsible for managing the application. The administrator can add new parking locations to the database and can also remove parking locations from the database.
* User: The user is the person who will be using the application. The user can search for parking near a specific location.
* Guest: The guest is a person who is not logged in to the application. The guest can only search for parking near a specific location.
* Monitor Bot: A monitor is a bot that will be monitoring a car park. The monitor will be updating the status of the car park.

## Use Case Descriptions

### Use Case: Find Parking

Description:

A user searches for parking near a specific location.

**Actors:**

* User

Trigger Event:

* A user wants to find parking near a specific location.

Preconditions:

* The user is logged in to the application.
* The website has permission to access the user’s GPS location.

Post conditions:

* The user is shown a list of parking locations near the location they searched for.

Main Flow:

1. The user details the location they want to find parking nearby:
   * by searching for a specific address in the search bar.
   * by clicking on a location on the map.
   * by clicking on a location on the list of parking locations.
   * using the current location of the user.
2. The application shows the user a list of parking locations near the location they searched for.

Alternative Flows:

* If the user does not have permission to access their GPS location, the user can search for a specific address in the search bar.

### Use Case: Register User

Description:

A user registers for an account on the application.

**Actors:**

* Guest user

Trigger Event:

* A guest user wants to register for an account on the application.

Preconditions:

* The guest user is not logged in to the application.
* The guest user has not registered for an account on the application.
* The guest has a valid email address.

Post conditions:

* A user account is created for the guest user.

Main Flow:

1. The guest user clicks on the “Register” button.
2. The guest user enters their details into the registration form.
3. The guest user clicks on the “Register” button.
4. The application creates a user account for the guest user.
5. The guest logs in to the application.

Alternative Flows:

* If the guest user enters an email address that is already registered to an account, the application will display an error message.

Use Case: Login User:

**Description:**

A user logs in to the application.

**Actors:**

* User

Trigger Event:

* A user wants to log in to the application.

Preconditions:

* The user is not logged in to the application.

Post conditions:

* The user is logged in to the application.

Main Flow:

1. The user clicks on the “Login” button.
2. The user enters their details into the login form.
3. The user clicks on the “Login” button.
4. The application logs the user in to the application.

Alternative Flows:

* If the user enters an incorrect username and password, the application will display an error message.
* If the user enters an username that is not registered to an account, the application will display an error message.
* If the user account is disabled, the application will display an error message.

### Use Case: Update Parking Lot Status

Description:

A monitor bot automatically updates the status of a parking lot.

**Actors:**

* Monitor

Trigger Event:

* A monitor updates the status of a parking lot.

Preconditions:

* The website application is running.
* The monitor is connected to the internet.
* The monitor has a valid API access token.

Post conditions:

* The status of the parking lot is updated.

Main Flow:

1. The monitor sends a PUT request to the application REST API.
2. The application updates the status of the parking lot.

Alternative Flows:

* If the monitor is not connected to the internet, the monitor will not be able to update the status of the parking lot.
* If the monitor API access token is invalid or has expired, the monitor will not be able to update the status of the parking lot.
* If the monitor sends an invalid request to the application REST API, the application will not update the status of the parking lot.
* If the parking lot does not exist in the database, the application will not update the status of the parking lot.

### Use Case: User changes password.

**Description:**

* A user changes their password.

**Actors:**

* User

**Trigger Event:**

* A user wants or is required to change their password.

**Preconditions:**

* The user is logged in to the application.

**Post conditions:**

* The user’s password is changed.

**Main Flow:**

1. The user clicks on the “Change Password” button.
2. The user enters their details into the change password form.
3. The user clicks on the “Change Password” button.

**Alternative Flows:**

* If the user enters an incorrect password, the application will display an error message.
* If the user enters a new password that does not meet the password requirements, the application will display an error message.

## Identifying the free/busy car parking spaces

A key design goal of the application is to find a low-cost, accurate, and scalable solution to identify if a car parking space is free or busy. To accomplish this goal, the application will use a combination of sensors and machine learning to identify if a car parking space is free or busy. The sensors will monitor the car parking space and will send data to a central server. The central server will then use a machine learning algorithm to identify if the car parking space is free or busy.

## Machine Learning / Artificial Intelligence (AI)

Machine learning is a subset of artificial intelligence (AI) that uses algorithms to learn from data and make predictions. Machine learning is a key component of the application as it will be used to identify if a car parking space is free or busy. The machine learning algorithm will be trained using data collected from the sensors. The machine learning algorithm will then be used to identify if a car parking space is free or busy.

## A look to the future

While developing the Perfect Parking application there was a major challenge when it came to incorporating parking sensors into the application. Due to budget and time constraints, it was determined that such sensors were beyond the scope of the project. While doing the project it was recognized that parking sensors would have been a valuable addition to the application but understood that this would require collecting large amounts of personal data from users. Since the project was not focused on data collection, the idea of the parking monitor came to light and so took priority over the parking sensors.

### Sensors

Overhead Cameras will watch the car parking space and will feed the video stream to a local client application. The local client application will use machine learning algorithms to identify if a car parking space is free or busy. The local client application will then send the status of the car parking space to the central server if it detects a change in the status of the car parking space.

### Collecting of data

When tech companies are building applications such as parking applications, they are given a budget by the parking company, and this enables them to put in these parking sensors or put in barriers to gather the real-time data for the users. This is where I face a big problem with gathering this data, since TUS carpark and other car parks in the city is monitored by another parking company called APCOA I am very limited to what data I can gather. Since I’m building this application on a very small scale gathering real-time data is going to be nearly impossible. One solution that I did think of would be to build my own parking space sensor using a raspberry pi and putting this down in a parking space in either my college or in the city, with doing this brings even more problems, these problems being:

* Permission must be sought from APCAO to allow a sensor to be placed on their parking premises.
* If I do get permission since the sensor would only be on a raspberry pi it could easily get damaged or stolen.
* I would only be able to build one sensor for one single parking spot which wouldn’t gather much real-time data for the users.

## Development limitations

This being a trial application, with a limited budget and permission problems, for the purpose of a university project, the client application will receive a video stream from a prerecorded local video file instead of a camera.

## Machine Learning Algorithms

* OpenCV <https://opencv.org/>
* Hough Line Transform <https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html>

## Database design

Diagram

Description automatically generated

Figure 3 - Database Design

The database for this project is composed of three tables: "User", "ParkingLotMonitor", and "ParkingLot".

The "User" table has three columns: "Id" (primary key), "Username", and "Password". This table stores the login information for users of the parking application.

The "ParkingLotMonitor" table has six columns: "Id" (primary key), "ParkingLotId" (foreign key), "ProbabilityParkingAvailable", "LastUpdated", "Status", and "DataTime". This table tracks the parking availability at each parking lot monitored by the application. The "ParkingLotId" column acts as a connection point, linking the "ParkingLotMonitor" table to the "ParkingLot" table.

The "ParkingLot" table has seven columns: "Id" (primary key), "Name", "Address", "Image", "Hours", "IsPaidParking", "Latitude", and "Longitude". This table stores information about each parking lot, including its name, address, image, hours of operation, and whether it is a paid parking lot. The "Latitude" and "Longitude" columns allow the parking application to display the parking lot location on a map.

The diagrams depict a relationship between the "ParkingLotMonitor" and "ParkingLot" tables using the "has" symbol. The "ParkingLotId" column in the "ParkingLotMonitor" table serves as a foreign key to link the two tables, allowing the application to track parking availability at each parking lot.

## User Parking Sequence diagram

A screenshot of a computer

Description automatically generated

Figure 4- User Parking Sequence Diagram

this is the sequence diagram of the process where a user is searching for parking near their location using the application. The user asks the app if there is parking available near their GPS location on Henry Street.

The app then queries multiple parking lot monitors, HenryStParkingLotMonitor, LowerHartstongeParkingLotMonitor, MallowStreetPart1ParkingLotMonitor, and MallowStreetPart2ParkingLotMonitor, to check if parking is available in each lot.

The HenryStParkingLotMonitor responds that parking is 97% available, the LowerHartstongeParkingLotMonitor responds 87%, MallowStreetPart1ParkingLotMonitor 65%, and MallowStreetPart2ParkingLotMonitor 45%.

Finally, the application then sends a response to the user indicating that there are 3 parking lots available near their location, with the names HenrySt, LowerHartstonge, and MallowStreetPart1.

# Implementation

## Object Recognition in Images

Object recognition in images is a popular computer vision task that involves detecting and localizing objects of interest within an image. This can be achieved using various techniques, such as feature extraction, machine learning, and deep learning. OpenCV is a popular library for computer vision and image processing that provides various tools and functions for performing object recognition.

The project implements object recognition using OpenCV. The project uses the Haar Cascade Classifier, which is a machine learning-based approach for object detection. The Haar Cascade Classifier works by detecting features in an image that are characteristic of the object being detected, such as edges, corners, and lines. These features are then used to classify the object. Haar Cascade Classifiers can be trained to recognize specific objects, such as faces, eyes, and cars, and the code is able to detect and localize these objects within an image.

The project code uses the Haar Cascade Classifier to detect and recognize different objects within an image. The project also implements various pre-processing techniques, such as image resizing and normalization, to improve the accuracy of the object detection.

## How Object Recognition and AI Is Used in Perfect Parking

The project makes use of the free and open-source OpenCV computer vision library to identify and track vehicles in designated parking spaces as well as to deliver real-time updates on parking spot availability. The programme overlays the designated parking spaces into the video, initialising them as available or occupied dependent on the presence of cars.

In the project, deep learning methods were used to train the object recognition system to identify and categorize parking spaces as occupied or vacant. This makes it a strong and adaptable tool for monitoring parking spaces, the program was able to learn and adjust to various lighting situations, car shapes and sizes, and other environmental parameters. The program was able to correctly identify when a car was present in a location by analyzing the average pixel intensity within the marked area after being trained on sizable datasets of labelled photos.

## Client and Server Architecture with Rest framework

### The Monitor

The Perfect Parking Server receives parking status data from client applications. A client monitor app is responsible for processing video and determining if parking is available. A proof-of-concept project by Olga Rocheeva was sourced on GitHub and built upon to work with Perfect Parking. (Rocheeva, 2018)

To setup a client, an administrator must mark out the spaces in an image of the video field before running the client application.

To determine whether a car is present in the spot, the client python file motion\_detector.py checks the average pixel intensity within the parking spots and comparing it to a threshold value. A location is regarded as available if the average intensity is below the threshold value and seen as occupied if it is above.

1. def detect\_motion(self):

2.     # ...

3.     coordinates = self.\_coordinates(p)

4.     logging.debug("coordinates: %s", coordinates)

5.

6.     rect = open\_cv.boundingRect(coordinates)

7.     logging.debug("rect: %s", rect)

8.

9.     new\_coordinates = coordinates.copy()

10.     new\_coordinates[:, 0] = coordinates[:, 0] - rect[0]

11.     new\_coordinates[:, 1] = coordinates[:, 1] - rect[1]

12.     logging.debug("new\_coordinates: %s", new\_coordinates)

13.

14.     # ...

15.

16.     mask = open\_cv.drawContours(

17.         np.zeros((rect[3], rect[2]), dtype=np.uint8),

18.         [new\_coordinates],

19.         contourIdx=-1,

20.         color=255,

21.         thickness=-1,

22.         lineType=open\_cv.LINE\_8)

23.

24.     mask = mask == 255

25.     self.mask.append(mask)

26.     logging.debug("mask: %s", self.mask)

27.     # ...

28.

Protecting Private Data (Useranems, passords)

The code in the server consists of five Python files: color.py, coordinates\_generator.py, drawing\_utils.py, motion\_detector.py, and main.py. The code provides functionality to generate coordinates for an image and detect motion in a video using OpenCV.

Let's look at the architecture of the Monitor code, starting with the entry point, main.py. This script handles command-line arguments using argparse, parses the YAML data file generated by coordinates\_generator.py and passes it to motion\_detector.py. If an image file is passed as an argument, it generates the YAML file with the coordinates. main.py then calls the detect\_motion() function of the MotionDetector class in motion\_detector.py.

The MotionDetector class is the main driver for motion detection. It reads frames from the video and compares them to the previous frame using cv2.absdiff(). If the difference is above a certain threshold, it marks the frame as containing motion and uses the cv2.findContours() function to find contours around the moving objects. It then loops through each contour and checks if it is inside any of the regions defined by the YAML file. If a contour is found inside a region, it sends an HTTP POST request to a specified URL using the requests library.

The CoordinatesGenerator class in coordinates\_generator.py is responsible for generating the YAML data file. It reads an image file and allows the user to click on four points to define a region of interest. It then writes the coordinates of the rectangle defined by those points to the YAML file. The class uses cv2.namedWindow() and cv2.setMouseCallback() to handle mouse events and updates the image displayed to the user with each mouse click.

The draw\_contours() function in drawing\_utils.py is a utility function for drawing contours and labels on an image. It takes an image and a set of coordinates, draws the contour around the coordinates, and places a label on the contour with a specified color, font, and thickness.

Finally, color.py contains colour constants that are used in other files.

In terms of architecture, the code follows a modular design pattern, with each file containing a set of related functions or classes. The main entry point is main.py, which coordinates the execution of the other files. The code uses several third-party libraries, including OpenCV, numpy, and requests. The MotionDetector class communicates with an external system using HTTP POST requests, making it easy to integrate the motion detection system with other systems.

### Server

The server-side code for this Django project consists of several files, including admin.py, apps.py, models.py, serializers.py, urls.py, and views.py.

admin.py registers the app's ParkingLot and ParkingLotMonitor models with Django's admin site.

apps.py defines the app's configuration, including its name and default auto field.

models.py defines the ParkingLot and ParkingLotMonitor models. The ParkingLot model has fields for id, name, address, hours, isPaidParking, latitude, longitude, image, and parking\_spaces. The ParkingLotMonitor model has fields for id, parkingLot, name, latitude,longitude, probabilityParkingAvailable, free\_parking\_spaces, dateTimeLastUpdated, status, and image.

Diagram

Description automatically generated

serializers.py defines the serializers used to convert the ParkingLot and ParkingLotMonitor models to JSON format for use in the app's API.

urls.py defines the app's URLs, including paths for the home page, parking lots list, parking lot detail page, user registration,

user login, user logout, and API endpoints for the ParkingLot and ParkingLotMonitor models.

views.py contains the logic for rendering the app's web pages, handling user input, and providing data to the app's API endpoints. It includes functions for rendering the home page, parking lot list, parking lot detail page, user registration, user login, and user logout pages. It also includes functions for handling API requests, including getting a list of all parking lots, getting details for a specific parking lot, and updating the parking lot monitor data.

Overall, the client-side code is responsible for providing a user-friendly interface for the app, handling user input and interactions, and communicating with the server-side code to retrieve and display data.

### How They Work Together

The client and server communicate with each other through the REST API service. When the client sends a request to the server, it includes information such as the endpoint to access, any data to send in the request body, and any headers to include with the request. The server-side code receives the request and processes it, querying the database or performing other operations based on the data included in the request. The server then sends a response back to the client, which includes a status code indicating whether the request was successful or not, any data to include in the response body, and any headers to include with the response. The client-side then processes the response, displaying the data in the website for probability of parking available so the user can view it. This cycle of request and response is how the client and server are linked together in a web application.

## Source Control and versioning

For source control and versioning, GitHub was utilized to manage the codebase for the project. GitHub was chosen because of prior experience using it during other studies, and it provided a reliable platform for version control and collaboration with the supervisor.

To keep the code organized and easily manageable, the repository was organised into different branches for each week of development. This allowed for easy upkeep and to keep track of changes made during each week and easily roll back to previous versions if needed.

Additionally, the supervisor was added as a collaborator on the repository, allowing him to view the progress and provide feedback on the code and documents. This facilitated effective communication and ensured that the project was aligned with the objectives.

One significant advantage of using GitHub was that it provided a safe and secure backup of the code. In the event of file corruption, it would be possible to pull down the last push request and continue the work without losing progress.

Overall, the use of GitHub as a source control and versioning tool allowed for effective management of the development of the Perfect Parking project, collaborate with the supervisor, and ensure the safety and integrity of the codebase.

## Tools Used

Tools:

* VS Code, a code editor that provides an excellent development environment.
* Anaconda, a package management, and deployment tool that made it easy to install and manage required libraries and dependencies.
* GitHub, a code repository that allowed for version control and collaboration with my supervisor.
* Microsoft Word, which was used to write the thesis.
* Canva, a graphic design platform used to create the project poster.

Languages and Frameworks:

* Django Python web framework, which allowed for rapid development of the project and easy maintenance.
* OpenCV, an open-source computer vision and machine learning software library, which was used for image processing and analysis

By utilizing these tools, languages, and frameworks, the project was able to be completed more efficiently, with greater accuracy and precision.

## Django Rest API

Django is a popular web development framework that is written in Python. It provides a set of tools and features that make it easy to build complex web applications quickly and efficiently. Django was created by Adrian Holovaty and Simon Willison in 2005, it features a vast collection of classes, libraries and modules that can be implemented in individual projects. With Django, you can create web applications that follow the Model-View-Controller (MVC) architecture, which helps to separate the different components of your application and make it easier to manage. Additionally, Django comes with a lot of built-in functionality, including an ORM for database interactions, an admin interface for managing site content, and a templating system for rendering HTML pages. Overall, Django is a powerful and flexible framework that is well-suited for building all kinds of web applications. (Johnson, n.d.)

### Perfect Parking with Django

The Perfect Parking application was created using the popular web framework Django for a variety of reasons.

Firstly, Django is a high-level web framework that follows the Model-View-Controller (MVC) architectural pattern, which promotes code organization and separation of concerns. This allows for the development of complex programmes with numerous components without compromising the maintainability of the code.

Secondly, Django provides a lot of built-in functionality out of the box, which saves time and effort during development. For example, Django includes an Object-Relational Mapping (ORM) system that allows developers to interact with databases using Python objects, as well as a robust authentication system for user management.

Thirdly, Django has an engaged community that actively supports the framework's growth and upkeep. This indicates that a wide variety of third-party packages and extensions are readily available and can increase the capabilities of the framework and speed up development.

Along with these benefits, Django's outstanding documentation, scalability, and security capabilities are some of the other benefits of adopting it for web development. Django is also open-source and free, which makes it available to a variety of developers and organisations.

Overall, the decision to use Django for the Perfect Parking application was based on its combination of ease of use, built-in functionality, and strong community support, which makes it a popular choice for building web applications of all sizes and complexities.

## Anaconda

Anaconda is a popular distribution of the Python programming language that is widely used for data science and scientific computing. It comes with a sizable number of pre-installed libraries and tools that are frequently used in these domains, including Jupyter Notebook, NumPy, Pandas, and Matplotlib.

Anaconda is designed to make it easy to set up and manage Python environments, which are essentially separate installations of Python with their own dependencies and libraries. When working on several projects with various requirements, this is especially helpful because it enables you to keep them separate from one another.

The Conda package manager, which lets you easily install, update, and manage additional software packages and libraries, is included with Anaconda in addition to Python and its libraries. When working with non-Python libraries that are necessary for your project, this can be helpful.

Utilising Anaconda has several benefits, one of which is how much easier it makes it to set up a Python environment for data research or scientific computing. It removes the need to individually install and configure each library, which can be a time-consuming and error-prone operation, by offering a pre-built distribution with many of the frequently used libraries already installed.

Overall, Anaconda is a robust and adaptable tool that is well-liked by those who work in data research and scientific computing. Researchers, developers, and data analysts all favour it because of how simple it is to use and the extensive library of tools that are already installed.

# Conclusion and Recommendations

## Conclusion

This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.

## Recommendations

* This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.
* This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.
* This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.
* This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.
* This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.
* This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.
* This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.
* This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs.

References

Anon., 2021. [Online]   
Available at: https://www.splunk.com/en\_us/data-insider/what-is-real-time-data.html

Anon., 2022. [Online]   
Available at: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

Anon., 2022. [Online]   
Available at: https://www.ul.ie/presidents-office/university-profile/facts-and-figures

Anon., n.d. [Online]   
Available at: https://techvidvan.com/tutorials/why-big-data/

Ashish, 2022. [Online]   
Available at: https://www.scienceabc.com/innovation/how-does-google-maps-know-about-traffic-conditions.html#:~:text=Google%20Traffic%20works%20by%20crowdsourcing,geographic%20location%20with%20the%20app.

Barnett, M., 2017. [Online]   
Available at: https://www.fourthsource.com/data/importance-real-time-data-five-reasons-need-22014

Botelho, B., n.d. *Big Data.* [Online]   
Available at: https://www.techtarget.com/searchdatamanagement/definition/big-data

Quellmalz, R., 2021. *6 surprising facts about parking you probably dont know.* [Online]   
Available at: https://www.spotparking.com.au/insights/facts-about-parking-you-probably-didnt-know

Glossary

|  |  |
| --- | --- |
|  |  |
| Term 1 | This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs |
| Term 1 | This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs |
| Term 1 | This chapter will begin by outlining the (cf. 1.1) for the purpose of writing a Report for a Project and outlining paragraphs |

1. Reflections
   1. Report Structure
2. Project Management
   1. Report Structure
   2. Code Style Guide
      1. Naming conventions
      2. Avoid magic constant numbers.
      3. Variable naming
      4. Methods
      5. Imports
      6. Comments
      7. Documentation
      8. Classes
      9. Spacing, Indentation
      10. Literals
3. Development Environment