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 with their support during my time at TUS.

Abstract

*This project is a Parking application for academic purpose. The aim of my project is to create a parking system that will replace the outdated systems and to help stop the widespread problem that is parking in our cities.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 Django framework structure and by implementing methods such map API’s 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](#_Toc132388007)

[Abstract iii](#_Toc132388008)

[Table of Contents iv](#_Toc132388009)

[Table of Figure viii](#_Toc132388010)

[Chapter 1 Introduction 9](#_Toc132388011)

[1.1 The academic objectives 9](#_Toc132388012)

[1.2 Problem Domain? 9](#_Toc132388013)

[1.3 Product title: a solution 9](#_Toc132388014)

[1.4 Objectives 9](#_Toc132388015)

[1.5 The Scope of the solution 9](#_Toc132388016)

[1.6 Report Structure 9](#_Toc132388017)

[Chapter 2 Materials 11](#_Toc132388018)

[2.1 Existing Data 11](#_Toc132388019)

[2.1.0 Others 11](#_Toc132388020)

[2.2 How we can choose 11](#_Toc132388021)

[2.2.1 Machine Learning 11](#_Toc132388022)

[2.3 Conclusion: The Need for a Software Solution 11](#_Toc132388023)

[Chapter 3 Project Management 12](#_Toc132388024)

[3.1 Weekly Meetings 12](#_Toc132388025)

[3.2 Source code management (SCM) 12](#_Toc132388026)

[3.3 Code Style Guide 12](#_Toc132388027)

[3.4 Collaboration Tools 12](#_Toc132388028)

[3.4.1 GitHub 12](#_Toc132388029)

[3.4.2 Microsoft Office Online 12](#_Toc132388030)

[Chapter 4 Data Analytic Methods 13](#_Toc132388031)

[4.1 Artificial Intelligence 13](#_Toc132388032)

[4.2 Categorization 13](#_Toc132388033)

[4.3 Estimation 13](#_Toc132388034)

[4.4 Machine Learning 13](#_Toc132388035)

[4.4.1 Garbage in, likely garbage out 13](#_Toc132388036)

[4.5 Working with Data Structures Object Orientated Programming 13](#_Toc132388037)

[4.6 Examples 13](#_Toc132388038)

[4.7 Conclusion 13](#_Toc132388039)

[Chapter 5 Data Analysis 15](#_Toc132388040)

[5.1 Introduction and focus 15](#_Toc132388041)

[5.2 Academic Aims 15](#_Toc132388042)

[5.2.1 Academic Requirements 15](#_Toc132388043)

[5.3 Functional Requirements 15](#_Toc132388044)

[5.4 Non-Functional Requirements 15](#_Toc132388045)

[5.5 Statistics 15](#_Toc132388046)

[Chapter 6 Results 16](#_Toc132388047)

[6.1 Project Plan: Priorities and Milestones 16](#_Toc132388048)

[6.1.0 The Data Structure 16](#_Toc132388049)

[6.1.1 Populating the System with Data 16](#_Toc132388050)

[6.1.2 Machine Learning 16](#_Toc132388051)

[6.1.3 Testing 16](#_Toc132388052)

[6.1.4 Paths to completion 16](#_Toc132388053)

[6.2 Data Structures 16](#_Toc132388054)

[6.3 System Architecture 16](#_Toc132388055)

[6.3.1 Object Identification 16](#_Toc132388056)

[6.4 Machine Learning 16](#_Toc132388057)

[6.5 Conclusion 16](#_Toc132388058)

[Chapter 7 Implementation 17](#_Toc132388059)

[7.1 Standards and Best Practice 17](#_Toc132388060)

[7.1.1 Object Orientated Programming 17](#_Toc132388061)

[7.1.2 Source Control and versioning 17](#_Toc132388062)

[7.2 Development Environment 17](#_Toc132388063)

[7.3 Tools Used 17](#_Toc132388064)

[Chapter 8 Conclusion and Recommendations 18](#_Toc132388065)

[8.1 Conclusion 18](#_Toc132388066)

[8.2 Recommendations 18](#_Toc132388067)

[References 19](#_Toc132388068)

[Glossary 20](#_Toc132388069)

[Appendix A Reflections 21](#_Toc132388070)

[A.1 Report Structure 21](#_Toc132388071)

[Appendix B Project Management 22](#_Toc132388072)

[B.1 Report Structure 22](#_Toc132388073)

[B.2 Code Style Guide 22](#_Toc132388074)

[B.2.1 Naming conventions 22](#_Toc132388075)

[B.2.2 Avoid magic constant numbers. 22](#_Toc132388076)

[B.2.3 Variable naming 22](#_Toc132388077)

[B.2.4 Methods 22](#_Toc132388078)

[B.2.5 Imports 22](#_Toc132388079)

[B.2.6 Comments 22](#_Toc132388080)

[B.2.7 Documentation 22](#_Toc132388081)

[B.2.8 Classes 22](#_Toc132388082)

[B.2.9 Spacing, Indentation 22](#_Toc132388083)

[B.2.10 Literals 22](#_Toc132388084)

[Appendix C Development Environment 23](#_Toc132388085)

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.

It is fair to say that 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. (Anon., 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 Domain?

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

1. Numbered Bullet list.
2. Numbered Bullet list
3. Numbered Bullet item.
   1. Numbered Bullet item.
   2. Numbered Bullet item.
4. Numbered Bullet list

## Product title: a solution

## Objectives

## The Scope of the solution

## Report Structure

This document has cover pages …

An Abstract

TOC and TOF are generated automatically.

The Chapters the following styles

Paragraphs are 12pt Aril Justified with 1.5-line spaces and 6pt before with 3 pt after.

# Literature Review

## Big Data and Realtime Data

### 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. (Anon., 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 actually very smart, they use a device that 6.6 billion people in the world have and that being smart phones. (Anon., 2022) 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.

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.

## Object Recognition and AI

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

## How we can choose

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

### Machine Learning

## Conclusion: The Need for a Software Solution

# 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.

# Data Analytic Methods

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

## Artificial Intelligence

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

## Categorization

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

## Estimation

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

## Machine Learning

### Garbage in, likely garbage out

## Working with Data Structures Object Orientated Programming

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

## Examples

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

## Conclusion

This chapter has outlined the …

A picture containing shape

Description automatically generated

Figure TUS Logo

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 - 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 near by:
   * 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.

## 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.

### 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 - Database Design

This database design consists of three tables: "User", "ParkingLotMonitor", and "ParkingLot".

* The "User" table has three columns: "Id" (Primary Key), "Username", and "Password".
* The "ParkingLotMonitor" table has six columns: "Id" (Primary Key), "ParkingLotId" (Foreign Key), "ProbabilityParkingAvailable", "LastUpdated", "Status", and "DataTime".
* The "ParkingLot" table has seven columns: "Id" (Primary Key), "Name", "Address", "Image", "Hours", "IsPaidParking", "Latitude", and "Longitude".

The diagrams shows a relationship between the "ParkingLotMonitor" and "ParkingLot" tables through the use of the "has" symbol. The "ParkingLotId" column in the "ParkingLotMonitor" table acts as a connection point, serving as a foreign key to link the two tables.

## User Parking Sequence diagram

A screenshot of a computer

Description automatically generated

Figure - 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.

## Introduction and focus

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

## Academic Aims

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

* Bullets
* Bullets

### Academic Requirements

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

## Functional Requirements

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

## Non-Functional Requirements

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

## Statistics

# Results

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

## Project Plan: Priorities and Milestones

### The Data Structure

### Populating the System with Data

### Machine Learning

### Testing

### Paths to completion

## Data Structures

## System Architecture

### Object Identification

## Machine Learning

## Conclusion

This chapter has outlined the …

# 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 is 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.

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

## Client and Server Architecture with Rest framework

### The Monitor

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 color 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 client-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.

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

The solutions presented in this chapter are the best practices and patterns of all those tried in various versions throughout the lifecycles of the systems defines in section 1.2.

## Development Environment

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

## Tools Used

This chapter has outlined the …

# 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

ul.ie., 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

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

"I bring order to chaos" - The Borg Queen, 2373

A few sentences about how the project was managed. A bit about the code, the document, the research, budget and timing, management frameworks and so on.

* 1. Report Structure
  2. Code Style Guide

"This appears to be a region of space that doesn't have many rules. But I believe we can learn something from the events that have unfolded. In a part of space where there are few rules, it's more important than ever that we hold fast to our own." – Captain Janeway, 2372

* + 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

1. Development Environment