Carleton University

Department of Systems and Computer Engineering

ITEC 5010 – Applied Programming I

Automated Parking Monitoring System with a Dynamic User Interface

FINAL PROJECT REPORT

Presented to:

Dr. Sima Soltanpour

School of Information Technology

Submitted by:

Nihal Kulkarni

101178458

Table of Contents

Acknowledgment

I would like to express my sincere gratitude and appreciation to Professor Sima Soltanpour for her invaluable support and guidance throughout the course. Her expertise in core programming areas has been instrumental in my understanding of the subject. I thank her for her constant availability to provide suggestions. The dedication to answering all my queries regarding course assignments and project has been immensely valuable, and I am profoundly grateful for the willingness to guide me through every step of the way. Moreover, the remarkable insights and guidance throughout the course project have been crucial to its successful completion. Professor's mentorship has not only helped me in this course but has also equipped me with the skills and knowledge required to excel in my future endeavours. I feel privileged to have had the opportunity to work with such an exceptional professor.

Abstract

Traffic problems are a major issue in many cities around the world. Congestion is one of the most common traffic problems, which occurs when there are too many vehicles on the road and the flow of traffic slows down or comes to a stop. Congestion can lead to delays, frustration, and increased pollution with infrastructure being one of the main issues. Many cities have outdated or inadequate infrastructure to support the growing number of vehicles on the road and this can lead to congestion, accidents, and other traffic problems. The automated parking spot detection and updation is a GUI-based project which provides an innovative solution to address the challenges of parking management in urban areas. The project employs sensors and computer vision technologies to detect and monitor the availability of parking spots in real-time, allowing drivers to quickly locate an available parking spot and reducing traffic congestion caused by drivers in search of a parking spot. The system can be integrated with a payment gateway, making parking more convenient for drivers. The project is expected to improve the efficiency of parking management and enhance the overall experience for drivers in areas such as shopping malls, airports, hospitals, and other public areas. The contents of the project report are summarized as follows. Section 1 gives the introduction to the project; Section 2 provides a problem statement followed by a problem analysis in Section 3. Section 4 provides specifications required for the project, section 5 gives insights into design strategy, and section 6 analyses test cases followed by results in section 7. Section 8 provides a future scope of the project and concludes with section 9.

Keywords:

GUI, Automated Parking System (APS), Parking Management, pandas, Tkinter, PIR Sensors, Infrared, Pylint.

1. Introduction

The automated parking spot detection and updation project is an advanced parking management system that uses modern technology to detect and monitor the availability of parking spots in real-time. This project is designed to reduce the time and effort drivers spend searching for an available parking spot and minimize traffic congestion caused by cars circling in search of parking. The system employs various sensors, such as ultrasonic, infrared, and cameras, to detect the presence of vehicles in parking spots [1]. These sensors relay information to a central computer system that processes the data and updates the availability of parking spots in real-time. The updated information is then displayed on digital display boards or mobile applications, enabling drivers to quickly locate available parking spots.

The project offers several benefits, including reduced traffic congestion, increased efficiency in parking management, and improved user experience for drivers. The system can also be integrated with payment gateways to allow drivers to pay for parking through the same mobile application or digital display board. The user interface will display parking lot information in a visually appealing manner, making it easy for parking administrators to monitor the parking lot's status. The automated parking spot detection project has the potential to revolutionize parking management in urban areas by improving the parking experience for drivers and reducing the environmental impact of traffic congestion caused by parking-related activities. The user interface will be designed to be intuitive and user-friendly, allowing parking administrators to access the information quickly and easily they need.

2. Problem Statement

The main objective of the automated parking spot detection project is to optimize parking management in urban areas. By utilizing advanced technology to detect and monitor the availability of parking spots in real time, the project aims to reduce the time and effort drivers spend searching for an available parking spot and minimize traffic congestion [2]. The project also aims to enhance the overall experience for drivers by providing real-time updates on available parking spots through digital display boards or mobile applications. Additionally, the system can be integrated with payment gateways, making the parking process more convenient for drivers.

3. Problem Analysis

3.1 Requirements Analysis:

Conducting a detailed analysis of the requirements and specifications of the project, including identifying the locations for implementing the system and the type of sensors and technologies required.

3.2 Design and Development:

Designing and developing the hardware and software components of the system, including sensors, computer vision algorithms, a central processing unit, and a display board or mobile application for users.

3.3 Installation and Integration:

Installing the hardware components of the system at the identified locations.

Integrating the software components with the hardware.

3.4 Testing and Validation:

Conducting extensive testing of the system to ensure that it performs as expected and meets the required standards for accuracy, reliability, and safety.

3.5 User Interface Development:

Designing and developing an intuitive user interface for the display board or mobile application that displays the availability of parking spots to users.

4. Specifications

The specification for this project is stated as follows.

4.1 Data Collection

4.1.1 PIR sensors

PIR (Passive Infrared) sensors are commonly used in APS to detect the presence of vehicles in parking slots and to display the number of available slots. These systems use a network of sensors that are installed in the parking lot and connected to a central control unit. Each parking slot is equipped with a PIR sensor that is mounted above the slot, facing downwards. When a vehicle enters or exits the slot, the PIR sensor detects the change in infrared radiation and sends a signal to the central control unit [3]. The control unit then processes the signal and updates the display to show the number of available parking slots.

In addition to PIR sensors, APS may also use other sensors, such as ultrasonic sensors or magnetic sensors, to detect the presence of vehicles. However, PIR sensors are often preferred because they are more reliable and consume less power than other types of sensors. Overall, PIR sensors play a crucial role in Automated Parking Systems by providing accurate and real-time information about the availability of parking slots. This helps drivers to find an available parking slot, reducing traffic congestion and improving the overall efficiency of the parking system quickly and easily.

4.1.2 Hardware Modules

The HC-SR505 hardware module is a popular choice for vehicle detection in Automated Parking Systems. Installed above each parking slot, the module's PIR sensor detects the presence of a vehicle through changes in infrared radiation [4]. The signal is then sent to the central control unit, which updates the display to show the number of available parking slots. The HC-SR505's low cost, low power consumption, and ease of use make it a reliable and cost-effective option for vehicle detection in Automated Parking Systems.

4.2 Libraries

4.2.1 Pandas

Pandas is a popular open-source data analysis library in Python that offers powerful data manipulation and analysis functionalities for handling large datasets. One of the key benefits of pandas is their ability to handle missing or incomplete data efficiently. Pandas can replace missing values, drop missing values, or interpolate the missing values using diverse statistical

methods. It can also be used in machine learning projects to pre-process and analyze data, as well as to visualize and explore data [5]. Overall, the pandas library is a powerful tool for data analysis in Python, offering efficient data manipulation, powerful data visualization, and easy handling of missing data

In Automated Parking Systems, pandas can be used to analyze and visualize data collected from various sensors and detectors, such as PIR sensors and ultrasonic sensors, to monitor parking slot occupancy and usage patterns. It can also help identify parking slot availability, optimize the allocation of parking spaces, and generate real-time reports and statistics. Overall, pandas will be used in the design, implementation, and management of Automated Parking Systems, helping to improve efficiency and reduce costs.

4.2.2 Tkinter

Tkinter is a standard GUI (Graphical User Interface) library in Python that provides a simple way to create windows, dialog boxes, and other graphical elements for desktop applications. It is included in most Python distributions and allows developers to create intuitive and user-friendly interfaces for their applications.

Tkinter offers a wide range of widgets, including buttons, labels, text boxes, menus, and more [6], that can be used to create a customized user interface. In Automated Parking Systems, Tkinter can be used to create a user interface for the system, allowing users to interact with the system through buttons, menus, and other visual elements. This can include functions such as selecting a parking slot, making a payment, and retrieving their vehicle.

5. Design Strategy

The flow chart process of an automated parking system based on the given conditions is as follows:

Start: The process begins with the start signal, indicating the initiation of the parking process.

Extract Sensor Data: The system extracts the sensor data from the parking area using various sensors like cameras, ultrasonic sensors, and infrared sensors.

Type of Vehicle - Car and Bike: Based on the sensor data, the system identifies the type of vehicle entering the parking lot, either a car or a bike.

Process Sensor Data: The system processes the sensor data by using advanced algorithms and machine learning techniques to analyze the data and identify the available parking spaces.

Update Occupancy Status: The system updates the occupancy status of the parking spots as either “occupied” or “vacant” in real time, depending on the vehicle's arrival and departure.

Display Occupancy status and set the refresh rate for UI: The system displays the occupancy status of the parking spots on the user interface (UI), allowing the drivers to quickly identify the vacant parking spots. The system also sets a refresh rate for the UI to update the occupancy status regularly.

Stop: The process ends with the stop signal, indicating the completion of the parking process.

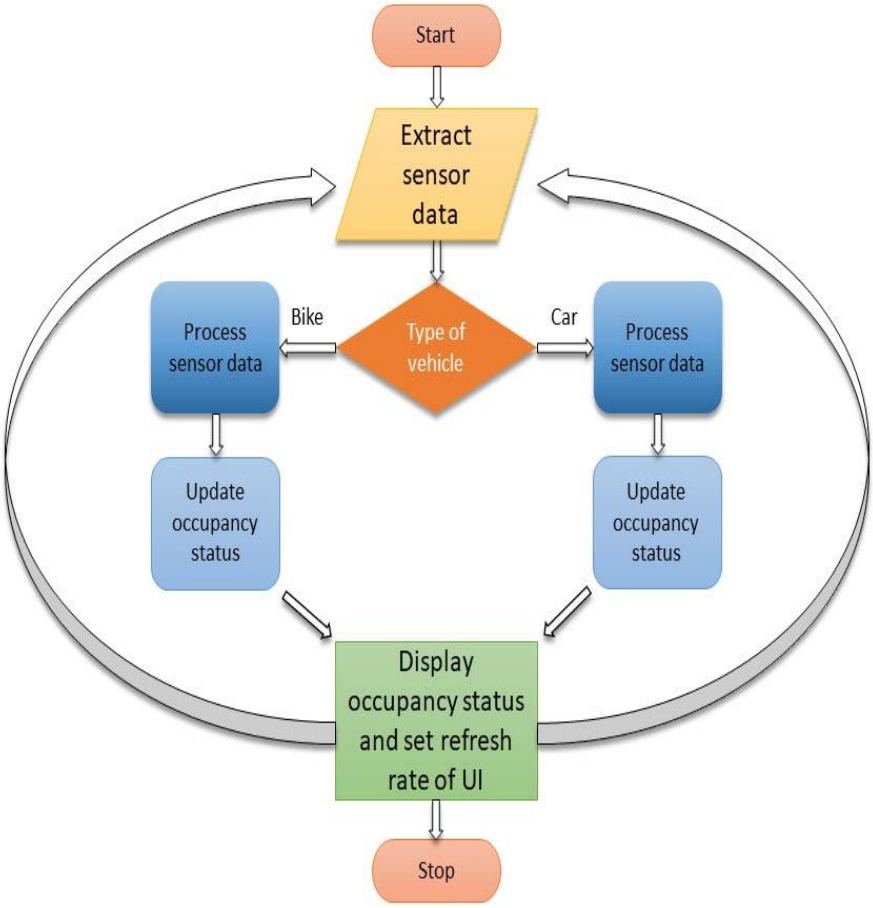

Fig. 1. Flow Chart for the System

6. Testing

The Automated Parking System testing is carried out by manipulating the CSV data files. The test Bikedata.csv and CarData.csv are two files that contain the ***Spot IDs*** which specify the number of spots in the parking area, and the ***Occupied*** tag specifies if the parking spot is occupied or available. In this project, we consider there is a total of 500 parking spots available for each set of cars and bikes. Each parking spot indicates a sensor in the lot. The 0's and 1's in the test dataset indicates spot is unoccupied or available and occupied or unavailable respectively.

6.1 Test Case 1: Manipulating Car Parking Spots

The random.csv file implemented and generated a random set of spots available and occupied. Fig. 1 shows parking information for cars. The default available spots are 269 out of 500, leaving 231 spots filled.

Fig. 2. Default Car Parking Information

When the data is manipulated in cardata.csv file by adding a 0, the GUI analyses the csv data and outputs the current number of available spots. The available spots now stand at 270. If 1 is added, then a spot is filled and now the available spot stands at 269.

Fig. 3. Manipulated Data with Updated Car Parking Information

6.2 Test Case 2: Manipulating Bike Parking Spots

The default available parking spaces for bikes stand at 225 out of 500. When the bike dataset is manipulated by adding a 0 it states the parking spot is now available. The GUI interface analyses the program and updates the interface with 226 available spots out of 500. Fig. 3 shows the default parking information and Fig. 4 shows the implementation of the bike data manipulation.

Fig. 4. Default Bike Parking Information

Fig. 5. Manipulated Data with Updated Bike Parking Information

7. Results

This project has been evaluated on a segment of the dataset from Indian traffic for parking lots. The dataset contains 1000 samples in residential areas, corporate hubs, shopping complexes, and public parking lots [7]. The dataset has been taken from different times of the day and evaluated performance for time and congestion in respective areas.

A graphical representation of the results was shown below in Figures 6 and 7 respectively. The net vehicle flow was taken for a typical working day in a metropolitan city. The first image represents the wait time of a vehicle looking for a vacant space to park. Much improvement is observed in shopping complex areas as the traffic flowing into malls is huge and the time to look for a spot to park the vehicle is high. With APS the time to look for a spot has been reduced by nearly 10x times and the effective time taken to park a vehicle was only considered rather than looking for the same. The next major improvement was in public parking spots as the

number of available slots is dynamic and the spot for any user varies for obvious reasons. The wait time has been reduced by approximately 5.5x times in public parking lots. Residential areas were considered for only visiting vehicles such as school buses, and guest vehicles as there would not be a fixed place for these vehicles to park. The wait time for each model of traffic varied with the time of the day and has shown drastic improvement in the evening time of the day. Overall, the wait time was reduced by 5x times on average in various types of parking units on a typical busy day.

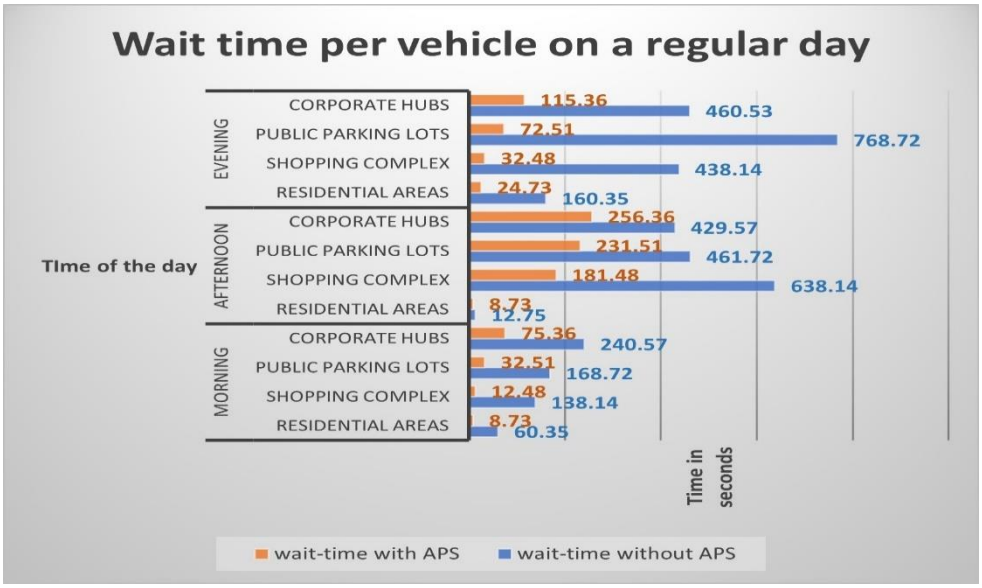

Fig. 6 Wait Time Comparison with and without APS

The project has also observed traffic congestion reduction in the areas close to these highly busy parking areas and the results for traffic congestion in the busy areas are shown. Residential areas and corporate hubs have reduced traffic congestion as there will be no wait time for cars or bikes to wait on the road. Since a greater number of vehicular movements is observed in the morning and evening times of the day where the typical workforce moves have seen a measurable amount of improvement in reduced traffic congestion. Overall, this approach has also reduced traffic jams and improved the quality of travel time on roads.

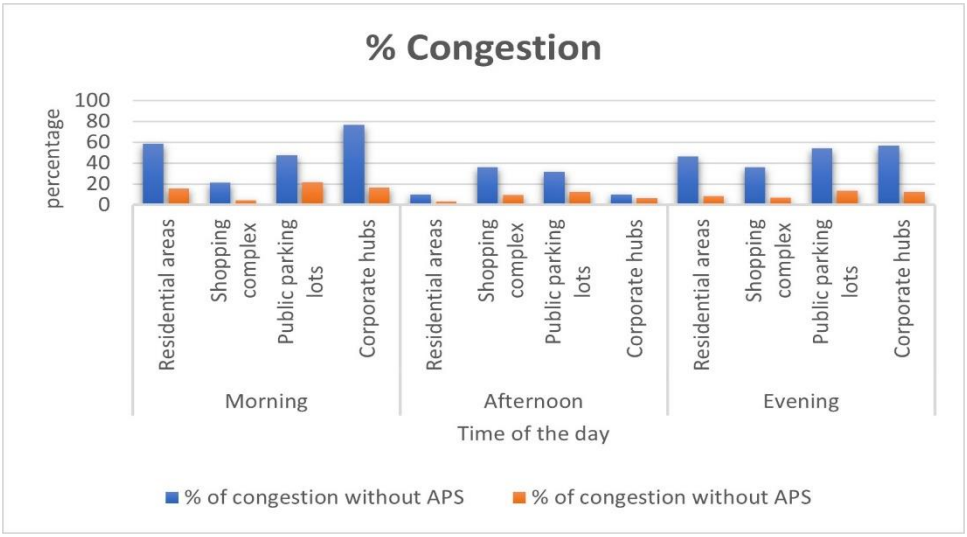

Fig. 7. Congestion comparison with & without APS

Pylint:

The project code quality and standard are verified by a Python code analyser called **Pylint** [8]. Pylint supports standard library functions and has an ecosystem of existing plugins. The current project code standard and quality are rated 10/10 based on the Pylint code analyser.

```
D:\Education\Python\Project>pylint parking_system.py --disable="global-statement"

--------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)
```

8. Future Scope

The future scope of automated parking systems is promising, with significant advancements expected in the areas of efficiency, safety, and sustainability. In the coming years, we can expect to see further developments in the technology and implementation of automated parking systems, as well as the integration of these systems with other technologies and infrastructure. One area of development is the integration of automated parking systems with smart city infrastructure. By connecting with other smart cities systems like traffic management, public transportation, and energy management, automated parking systems can contribute to the creation of more efficient and sustainable cities [9]. For example, by optimizing parking utilization, automated parking systems can reduce traffic congestion, improving the flow of traffic and reducing emissions.

Another area of development is the use of advanced machine learning algorithms and predictive analytics to optimize the parking experience for drivers [10]. By analyzing historical parking data and identifying patterns, automated parking systems can predict parking demand and optimize the utilization of available parking spaces. This can reduce the time and effort required to find a parking spot, improving the overall parking experience for drivers. By using advanced sensors and cameras, these systems can monitor the parking area and identify potential safety hazards, reducing the risk of accidents and collisions. Moreover, the integration of safety features like emergency stop mechanisms and fail-safes can further improve the safety of these systems. Overall, these systems can provide an optimized parking experience while contributing to the creation of more efficient and sustainable cities.

9. Conclusion

Automated parking systems have several technical features that contribute to their positive impact on parking management. These systems use advanced technologies like sensors, cameras, and algorithms to optimize space utilization, reduce wait times, and improve the overall parking experience for drivers. This data is then processed by advanced algorithms and machine learning techniques to analyze the data and identify the available parking spaces [11]. By using sensors and algorithms, the system can quickly identify available parking spaces and direct drivers to those spaces, minimizing the time and effort required to find a parking spot. In conclusion, the technical features of automated parking systems contribute to their positive impact on parking management by optimizing space utilization, reducing wait times, and improving the overall parking experience for drivers. By utilizing advanced technologies like sensors, cameras, and algorithms, these systems provide an efficient, streamlined, and sustainable solution to parking management.

10. References

[1] BR Kinzey, MA Myer, “Use of Occupancy Sensors in Parking Lot and Garage Applications”, U.S Department of Energy, Washington.

[2] Faris Alshehri, A.H.M Almawgani, “Smart Parking System for Monitoring Cars and Wrong Parking”, IEEE Communications, Saudi Arabia, 2019.

[3] Brian Donovan, Yanning Li, “Vehicle detection and speed estimation with PIR sensors”, Proceedings of the 14th International Conference on Information Processing in Sensor Networks, 2015

[4] Jeff S. Shamma, Christian Claudel, “Vehicle Classification and Speed Estimation Using Combined Passive Infrared/Ultrasonic Sensors”, IEEE Transactions on Intelligent Transportation Systems, Volume: 19, Issue: 5, May 2018

[5] https://pandas.pydata.org/docs/user_guide/index.html

[6] https://docs.python.org/3/library/tkinter.ttk.html

[7] https://www.kaggle.com/datasets/blanderbuss/parking-lot-dataset/code

[8] https://pypi.org/project/pylint/

[9] D.J. Bonde; Rohit Sunil Shende; Akshay Sambhaji Kedari, “Automated car parking system commanded by Android application”, International Conference on Computer Communication and Informatics, 2014

[10] Sunethra B, Sreeya C, Dhannushree U, “A Systematic Parking System Using bi-class Machine Learning Techniques”, Proceedings of the International Conference on Sustainable Computing and Data Communication Systems (ICSCDS-2022)

[11] Pampa Sadhukhan, “An IoT-based E-parking system for smart cities”, International Conference on Advances in Computing, Communications, and Informatics (ICACCI), 2017

11. Appendix A: Automated Parking System Program Code

```
"""Module for Parking system"""
from tkinter import Tk,Label
import pandas as pd
import csv
from datetime import datetime

CSLOTS_OCCUPIED=0
CSLOTS_EMPTY=0
BSLOTS_OCCUPIED=0
BSLOTS_EMPTY=0
curr_bmtime=float('inf')
curr_ctime=float('inf')
Window = Tk()
#now = datetime.now()

def readcardata():
    """Method to read car slots"""
    global CSLOTS_EMPTY
    global CSLOTS_OCCUPIED
    with open("Cardata.csv","r",encoding="utf8") as cardata:
        cars = csv.DictReader(cardata)
        for car in cars:
            if car["Occupied"] == "1":
                CSLOTS_OCCUPIED=CSLOTS_OCCUPIED+1
            if car["Occupied"] == "0":
                CSLOTS_EMPTY=CSLOTS_EMPTY+1
        cardata.close()

def readbikedata():
    """Method to read bike slots"""
    global BSLOTS_EMPTY
    global BSLOTS_OCCUPIED
    with open("Bikedata.csv","r",encoding="utf8") as bikedata:
        bikes = csv.DictReader(bikedata)
        for bike in bikes:
            if bike["Occupied"] == "1":

            if bike["Occupied"] == "1":
                BSLOTS_OCCUPIED=BSLOTS_OCCUPIED+1
            if bike["Occupied"] == "0":
                BSLOTS_EMPTY=BSLOTS_EMPTY+1
        bikedata.close()

def track_parking():
    """Method to track time"""
    now = datetime.now()
    curr_time = now.strftime("%H:%M:%S")
    time_val = Label(Window)
    time_val.config(text=curr_time)
    time_val.grid(row=0,column=5)
    Window.after(1000, track_parking)


def parkingdata_ui():
    """Method to define graphical window"""
    global BSLOTS_EMPTY,BSLOTS_OCCUPIED,CSLOTS_EMPTY,CSLOTS_OCCUPIED
    Window.geometry("600x250")
    headlabel = Label(Window,text="Parking Information",font=20)
    carlabel = Label(Window,text="Cars",font=18)
    bikelabel = Label(Window,text="Bikes",font=18)
    car_available = Label(Window, text="Spots Empty:",padx=30)
    car_ava_count = Label(Window,text = "")
    car_ava_count["text"] = str(CSLOTS_EMPTY)
    car_filled = Label(Window,text="Spots Filled:")
    car_fil_count = Label(Window,text="")
    car_fil_count["text"] = str(CSLOTS_OCCUPIED)
    car_total = Label(Window,text="Total Spots:")
    car_count = Label(Window,text="")
    car_count["text"] = str(CSLOTS_EMPTY+CSLOTS_OCCUPIED)

    bike_available = Label(Window, text="Spots Empty:")
    bike ava count = Label(Window,text="")
```

```
    bike_ava_count["text"] =str(BSLOTS_EMPTY)
    bike_filled = Label(Window,text="Spots Filled:")
    bike_fil_count = Label(Window,text="")
    bike_fil_count["text"] = str(BSLOTS_OCCUPIED)
    bike_total = Label(Window,text="Total Spots:")
    bike_count = Label(Window,text="")
    bike_count["text"] = str(BSLOTS_EMPTY+BSLOTS_OCCUPIED)

    headlabel.grid(row=0,column=2)
    carlabel.grid(row=1,column=1)
    car_available.grid(row=2,column=0)
    car_ava_count.grid(row=2,column=1)
    car_filled.grid(row=3,column=0)
    car_fil_count.grid(row=3,column=1)
    car_total.grid(row=4,column=0)
    car_count.grid(row=4,column=1)

    bikelabel.grid(row=1,column=4)
    bike_available.grid(row=2,column=3)
    bike_ava_count.grid(row=2,column=4)
    bike_filled.grid(row=3,column=3)
    bike_fil_count.grid(row=3,column=4)
    bike_total.grid(row=4,column=3)
    bike_count.grid(row=4,column=4)
    CSLOTS_OCCUPIED=0
    CSLOTS_EMPTY=0
    BSLOTS_OCCUPIED=0
    BSLOTS_EMPTY=0

def parkingdata():
    """Method to update graphical window every 5 seconds"""
    readbikedata()
    readcardata()
    track_parking()
    parkingdata_ui()
    Window.title("Parking System")
    Window.after(5000, parkingdata)
    Window.mainloop()

def main():
    """main method to execute code"""
    parkingdata()

if __name__=="__main__":
    main()
```