

**Autonomous Parking Bays**

By

Antipas Immanuel Chinodakufa

(H190317J)

HIT400 Capstone project Submitted in Partial Fulfilment of the

Requirements of the degree of

Bachelor of Technology

In

**Software Engineering**

In the

**School of Information Sciences and Technology**

Harare Institute of Technology

Zimbabwe

Supervisor

Mr W. Makondo

…………………………………………………………………..

May 2023

**Certificate of Declaration**

This is to certify that work entitled “Autonomous Parking Bays “ *is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology (Hons) in Software Engineering ,Harare Institute of Technology .It is further certified that no part of research has been submitted to any university for the award of any other degree .*



(Supervisor) Signature…………………………….. Date……………………….

(Mentor) Signature…………………………….. Date………………………

(Chairman) Signature……………………………….. Date………………………..

**Abstract**

This project aims to design and implement an autonomous parking system that uses automatic number plate recognition (ANPR) and a payment platform to manage parking services for any owner of parking space in Zimbabwe. The current parking system relies on manual processes and human intervention, which are prone to errors, frauds, inefficiency, inconvenience, and high costs. The proposed parking system uses a camera and an optical character recognition (OCR) algorithm to scan the number plate of a car and start a timer when it enters a parking bay. The system also queries a database for the customer account associated with the number plate and updates it with the parking cost and records the car service and fee information. The customer can pay for the parking using PayNow, a third-party payment platform that is integrated with the payment system. The reporting system retrieves the car service and fee information from the database and generates a report of the parking activity. The objectives of this project are: 1) to detect a number plate of a vehicle on a parking bay; 2) to record the duration parked by a vehicle; 3) to calculate the parking cost based on the duration; and 4) to allow a customer to pay the parking fee after parking. The proposed parking system is expected to reduce errors and frauds, improve efficiency and speed, enhance convenience and safety for customers, lower costs and labour, and provide accurate and reliable data for reporting and analysis. The project will use appropriate software tools and techniques to design, develop, test, and implement the proposed parking system.

**Preface**

This project is about designing and implementing an autonomous parking bay system for supermarkets in Zimbabwe. The system aims to improve the parking experience for customers and reduce the costs and inefficiencies of manual parking systems.

The motivation for this project stems from the observation that parking systems in Zimbabwe are human intensive and prone to errors, frauds, and corruption. Supermarket customers often have bad parking experiences that affect their satisfaction and loyalty. Moreover, manual parking systems do not provide accurate and reliable data for reporting and analysis.

The main research question of this project is: How can an autonomous parking bay system improve the parking service quality and efficiency for supermarkets and other parking space providers in Zimbabwe? To answer this question, the project follows these objectives:

* To detect a number plate of a vehicle on a parking bay.
* To record the duration parked by a vehicle, from when it enters to when it leaves the parking bay.
* To calculate the parking cost based on the duration parked and the parking rate.
* To allow a customer to pay the parking fee after parking using PayNow, a third-party payment platform.

The main methods or approaches used in this project are:

* Image extraction using OpenCV and a webcam
* Image detection using OpenCV and pre-trained model.
* Image segmentation and recognition using EasyOCR
* Duration measurement using Python
* Payment integration using Django framework and PayNow test API
* Database management using PostgreSQL

The main results or contributions of this project are:

* A prototype of an autonomous parking bay system that can perform automatic number plate recognition (ANPR) on Zimbabwean number plates, measure and display the duration of parking, calculate and collect the parking fees, and generate reports of the parking activity.
* A comparison of the performance and accuracy of different image detection algorithms (TensorFlow vs OpenCV) and OCR tools (EasyOCR vs Tesseract) for ANPR.
* A user evaluation of the usability and satisfaction of the autonomous parking bay system.

The limitations or challenges of this project are:

* The system requires a stable internet connection and power supply to function properly.
* The system may not be able to detect or recognize some number plates due to poor lighting, dirt, damage, or occlusion.
* The system may not be compatible with some payment methods or platforms other than PayNow.
* The system may face legal or ethical issues regarding data privacy and security.

This project was completed in May 2023 as part of the Bachelor of Technology in Software Engineering degree at the Harare Institute of Technology. I would like to acknowledge the following sources of support or assistance for this project:

- My supervisor, Mr. Makondo, for his guidance and feedback throughout the project.

- My classmates, for their help with testing and debugging the system.

- My family and friends, for their encouragement and moral support.

**Acknowledgement**

I would like to express my sincere gratitude to the following people who have supported me throughout this project:

My supervisor, Mr. Makondo, for his valuable guidance, feedback, and encouragement. He constantly checked up on us, enquired on our progress and demanded continuous improvement. He was a true mentor

Kelvin Zawala, for providing me with a stable and power internet connection that enabled me to experiment with different software and tools. He also assisted me with some technical problems and shared his insights and suggestions.

My family and friends, for their constant support and motivation. They have been very understanding and supportive of my efforts and challenges. They have also cheered me up and inspired me to keep going.

**Dedication**

I dedicate this project to my parents, who brought home my first laptop at age 12, obliviously igniting a passion. They gave me the freedom to pursue my passions and they have been a constant support throughout. I hope to make them proud.

I also dedicate this project to my mentor, Mr. Makondo, who has been a seeker of continuous improvement. He challenged me and he sought to bring out the potential he saw. He was the perfect mentor, he was a great role model.

Finally, I dedicate this project to all the people who have helped me along the way. The friends who constantly checked up on my progress, sometimes wake me up at night better than an alarm and keep me up at night better than coffee. They have made this project possible and enjoyable.

**CONTENTS**

| **Chapter/Section** | **Title** | **Page** |
| --- | --- | --- |
| **Chapter One** | **Introduction** | **1** |
| 1.1 | Background | 1 |
| 1.2 | Problem Statement | 1 |
| 1.3 | Objectives | 2 |
| 1.4 | Hypothesis | 2 |
| 1.5 | Justification | 3 |
| 1.6 | Proposed Tools | 3 |
| 1.7 | Feasibility Study: Technical, Economic & Operational | 4 |
| 1.8 | Project Plan – Time Plan, Gantt chart | 6 |
| **Chapter Two** | **Literature Review** | **8** |
| 2.1 | Introduction | 8 |
| 2.2 | Related Work | 8 |
| 2.3 | Conclusion | 10 |
| **Chapter Three** | **Analysis** | **11** |
| 3.1 | Information Gathering Tools, Description of System | 11 |
| 3.2 | Data Analysis - Using UML diagrams (existing system), DFDs of existing system | 12 |
| 3.3 | Functional Analysis of Proposed System – Functional and Non-Functional Requirements, User Case Diagrams | 15 |
| **Chapter Four** | **Design** | **20** |
| 4.1 | Systems Diagrams – Using UML diagrams, DFD diagrams, Activity diagrams | 20 |
| 4.2 | Architectural Design – hardware, networking | 25 |
| 4.3 | Database Design – ER diagrams, Normalized Databases | 25 |
| 4.4 | Program Design – Class diagrams, Sequence diagrams, Package diagrams, Pseudocode | 32 |
| 4.5 | Interface Design – Screenshots of user interface | 35 |
| **Chapter Five** | **Implementation & Testing** | **38** |
| 5.1 | Pseudocode of major modules /Sample of real code can be written here | 38 |
| 5.2 | Software Testing – Unit, Module, Integration, System, Database & Acceptance Testing | 41 |
| 5.3 | Installation & deployment, Maintenance | 49 |
| **Chapter Six** | **Conclusion and Recommendations** | **51** |
| 6.1 | Results | 51 |
| 6.2 | Summary | 59 |
| 6.3 | Recommendations | 61 |
| 6.4 | Future Works | 61 |
| **Bibliography (References)** |  | **63** |
| **Appendices** |  | **66** |
| A | Sample code | 66 |
| B | Research papers (technical paper) | 66 |

**CHAPTER ONE**

**1.1 Background**

Parking systems all over Zimbabwe are human intensive and with all human intensive systems, they tend to err, miscalculate and are corrupt. Supermarket customers all over Zimbabwe have bad parking experiences coupled to every shopping experience and chances are they never come back. Such poor service provision will cause customers to find and favour alternative franchises out of city or go for a different supermarket brand with better services.

Autonomous parking bays allows for post parking payments for parking space owners selling their parking space. Autonomous parking bays works by performing ANPR (Automatic Number Plate Recognition) once a car pulls into a parking bay. It measures the length of that number plate detection hence measuring the time spent in parking. For 2 hour limits, it alerts the supermarket and measures can be taken. However for paid parking, an integrated payment system will allow for post parking payments. By implementing autonomous parking bays, supermarkets will retain their customers and make sure that there is always parking space for them.

**1.2 Problem statement**

The current parking system relies on manual processes and human intervention to identify cars and licence plates, collect payments, produce receipts, and keep track of time. This system has several drawbacks:

* It is prone to human errors and frauds
* It is inefficient and time-consuming
* It is inconvenient and unsafe for customers
* It is costly and labour-intensive
* It does not provide accurate and reliable data for reporting and analysis

The proposed parking system uses an automated detection system that scans the number plate using a camera and an OCR algorithm and starts a timer. The system also queries a database for the customer account associated with the number plate and updates it with the parking cost and records the car service and fee information. The customer can pay for the parking using PayNow, a third-party payment platform that is integrated with the payment system. The reporting system retrieves the car service and fee information from the database and generates a report of the parking activity.

The proposed parking system has several advantages over the current system:

* It reduces errors and frauds
* It improves efficiency and speed
* It enhances convenience and safety for customers
* It lowers costs and labor
* It provides accurate and reliable data for reporting and analysis

Therefore, the proposed parking system is a better solution for managing parking services than the current system. The goal of this project is to design, develop, test, and implement the proposed parking system using appropriate software tools and techniques.

**1.3 Objectives**

1. To detect a number plate of a vehicle on a parking bay.
2. To record the duration parked by a vehicle that is, from when a car is in site to when it leaves the parking bay.
3. To calculate the parking cost from the moment its number plate is dictated to the moment it leaves a parking bay.
4. To allow a customer to pay the parking fee after parking.

Scope of the project

The scope of the project is a system that can be used in a supermarket parking bay to either enforce 2 hour limit for free parking or calculate and collect parking fees for paid parking. The system will perform ANPR on Zimbabwean number plates. It will measure and display duration of parking as well as to calculate cost of parking. Furthermore, it will allow for post payment of parking fees incurred.

**1.4 Hypothesis**

The hypothesis is that an autonomous parking system will outperform a human intensive system in finance and efficiency. More so, the hypothesis is that object detection will be continuous for as long as the object is in sight because any break in detection will cause discrete time measurements and cost calculation will be inaccurate.

**1.5 Justification**

The current parking systems in Zimbabwe tend to err, miscalculate and are corrupt in their provision of parking and collection of payments. Autonomous parking bays will correct these three human cause by removing the human all together through the use of computer vision. Furthermore, removing the human, cuts down costs and improves quality of service by providing an efficient parking system.

When compared to existing system the following justifications can be made:

- The level of automation and intelligence of the system. The system is mostly automated and intelligent, as it uses a camera, a timer, a database, a payment platform, and a reporting system to perform the parking functions. Other systems may rely more on human intervention, such as manual ticketing, validation, payment, and monitoring.

- The cost and complexity of the system. The system may have a higher initial cost and complexity than other systems, as it requires more hardware, software, and integration components. However, the system may also have lower operational and maintenance costs than other systems, as it reduces human errors, frauds, and inefficiencies.

- The user experience and satisfaction of the system. The system may offer a better user experience and satisfaction than other systems, as it provides faster, easier, and more secure parking services. The system may also provide more transparency and accountability than other systems, as it records and reports the car service and fee information.

**1.6 Proposed tools**

- Image extraction – open cv & webcam

- Image detection – Tensorflow /open cv

- Image segmentation – easyOCR

- Image recognition –EasyOCR

- Duration measurement – python

- Payment integration – Django framework and Paynow test API

- Database – PostgreSQL

- Frontend – HTML, CSS and JS

**1.7 Feasibility study**

Technical feasibility

The following are the tools used by system

Webcam, Open CV, Tensorflow, EasyOCR, Django framework, Paynow test api

PostgreSQL, HTML, CSS and JS

The tools are freely available and technical skills to use them are sharp enough. Time to implement is sufficient enough for production of a minimum viable product. The system will be hosted in-house to begin and the system will be scaled accordingly as system grows.

Therefore, the autonomous parking bay system is technically feasible.

Economic feasibility

Development cost, the average cost of developing a parking system app is around $50,000 to $100,000. Assuming the system is a medium-sized project with moderate complexity and features, I will estimate the development cost to be $75,000.

Time: Assuming your system follows a waterfall model of software development, I will estimate the time required for each phase as follows: Planning: 2 weeks Analysis: 4 weeks Design: 6 weeks Implementation: 8 weeks Testing: 4 weeks Deployment: 2 weeks Maintenance: ongoing The total time required for the project is 26 weeks or about 6 months.

Hardware and infrastructure costs the hardware and infrastructure costs for a smart parking system may include the following components Cameras: $100 to $200 per unit, Servers: $500 to $1000 per unit, Network devices: $100 to $500 per unit, Software licenses: $1000 to $5000 per year, Assuming your system requires, 1000 cameras, 10 servers, 20 network devices, and 5 software licenses, I will estimate the hardware and infrastructure costs to be $150,000.

Cash flows the annual revenue for a parking system in Zimbabwe may range from $1.1 million to $1.8 million. Assuming the system charges $5 per hour for parking and has an average occupancy rate of 70%, I will estimate the annual cash inflow to be $1.5 million. Assuming your system has an annual operating cost of 10% of the cash inflow and a depreciation rate of 20% of the hardware and infrastructure costs, I will estimate the annual cash outflow to be $300,000. The net cash flow is the difference between the cash inflow and the cash outflow, which is $1.2 million.

Using these values and assumptions

1. Cost Benefit Analysis: The cost benefit analysis compares the total costs and benefits of the project over its lifetime. Assuming your system has a lifetime of 10 years and a discount rate of 10%, I will calculate the present value of the costs and benefits as follows: Present value of development cost = $75,000 / (1 + 0.1)^0 = $75,000 Present value of hardware and infrastructure costs = $150,000 / (1 + 0.1)^0 = $150,000 Present value of operating costs = ($150,000 / 0.1) x (1 - 1 / (1 + 0.1)^10) = $837,600 Present value of depreciation costs = ($30,000 / 0.1) x (1 - 1 / (1 + 0.1)^10) = $167,760 Present value of cash inflows = ($1.5 million / 0.1) x (1 - 1 / (1 + 0.1)^10) = $8.37 million Total present value of costs = $75,000 + $150,000 + $837,600 + $167,760 = $1.23 million Total present value of benefits = $8.37 million Net present value of benefits = $8.37 million - $1.23 million = $7.14 million Benefit-cost ratio = $8.37 million / $1.23 million = 6.81
2. Return on Investment: The return on investment measures the profitability and efficiency of the project or investment. ROI is calculated by dividing the net benefit or net profit by the total cost or total investment. ROI is usually expressed as a percentage or a ratio. Assuming your system has a lifetime of 10 years and a discount rate of 10%, I will calculate the ROI as follows: ROI = ($7.14 million / ($75,000 + $150,000)) x 100% = 4753%
3. Net Present Value: The net present value measures the present value of all future cash flows generated or saved by the project or investment. NPV is calculated by discounting all future cash flows by a certain interest rate or discount rate that reflects the opportunity cost of capital. NPV is usually expressed in monetary terms. Assuming your system has a lifetime of 10 years and a discount rate of 10%, I will calculate the NPV as follows: NPV = -$75,000 - $150,000 + ($1.2 million / 0.1) x (1 - 1 / (1 + 0.1)^10) = $7.14 million
4. Payback Period: The payback period measures how long it takes for a project or investment to recover its initial cost or break even. Payback period is calculated by dividing the initial cost by the annual cash flow or by adding up the cumulative cash flow until it equals the initial cost. Payback period is usually expressed in years or months. Assuming your system has a lifetime of 10 years and a discount rate of 10%, I will calculate the payback period as follows: Payback period = ($75,000 + $150,000) / $1.2 million = 0.19 years or 2.3 months

Using these values and assumptions, I have calculated the following metrics for the system:

* Cost Benefit Analysis: Net present value of benefits = $7.14 million Benefit-cost ratio = 6.81
* Return on Investment: ROI = 4753%
* Net Present Value: NPV = $7.14 million
* Payback Period: Payback period = 0.19 years or 2.3 months

These metrics indicate that your system is highly profitable, efficient and attractive as an investment in Zimbabwe. The system has a positive net present value, a high return on investment, a high benefit-cost ratio and a very short payback period. The system also adds value to the organization and exceeds its required rate of return.

Operational feasibility

The system is designed to be autonomous that means very little human interference is needed and this reduces cost by cutting manpower, it reduces human error which is a major collapse of many systems. The system is designed to have a high degree of efficiency and accuracy in managing parking times and costs taking this load from the human or organization. The system can be easily scaled if number of bays increased by just increasing camera modules.

Therefore, the autonomous parking bay system is operational feasible.

**1.7 Project plan**

Table 1 Gantt chart

2023

2022

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Activities** | **Aug** | **Sept** | | **Oct** | **Nov** | **Dec** | **Jan** | **Feb** | **Mar** |
| Proposal |  |  | |  |  |  |  |  |  |
| Supervisor Consultation |  |  | |  |  |  |  |  |  |
| Quick design |  |  |  |  |  |  |  |  |  |
| Build first prototype |  |  | |  |  |  |  |  |  |
| Evaluation and refining prototype |  |  | |  |  |  |  |  |  |
| implementation |  |  | |  |  |  |  |  |  |

**CHAPTER 2**

**Introduction**

Automatic Number Plate Recognition system (ANPR) is a way of identifying vehicles by their licence plates using image processing. ANPR is useful for managing parking areas because each vehicle has a unique licence plate that can be used for identification [1]. ANPR also has other uses and examples are controlling the border, measuring time for a journey, controlling access to private property, highway tolling and monitoring traffic on the roads. [2].

ANPR has four main stages which are image acquisition, number plate detection, character segmentation and character recognition [3] [4] [5] [7] [8]. ANPR does the following process:

i. Image taken from camera

ii. Image pre-processed

iii. Number plate detection

iv. Character segmentation

v. Character recognition

vi. Output of character.

[3] [5] [7]

Stage 1 – image acquisition – this is where a camera gets a video or an image and in case of video, it is changed to frames in a pre-processing step. It is also where grey scale conversion and noise filtering happen to make number plate recognition better [5]. In [7], pre-processing has 4 steps, grey scaling, noise removal, edge detection and dilation.

Stage 2 – number plate detection – This is how to get the licence plate area by cutting vehicle image for later recognition system to check [6]. Reducing processing time for licence plate detection is done by only processing pixel with licence plate features like colour, size, edge boundary and presence of character feature [4]. Licence plates have high contrast areas and these yellow and black or white and black areas help with licence plate recognition. [5][12].

Stage 3 – character segmentation – This is splitting the licence plate image again and again until every character is its own different sub-image [4].

Stage 4 – character recognition – This is where the characters in sub images made in segmentation are recognized. In [4], four methods are explained, ANN, Template matching, Tesseract OCR engine and Support Vector Machine (SVM). Template matching is the only recognition tool used in [5] [21]. On the other side, [11] used SVM and ANN as OCR tools to a 90% character recognition in wet condition. But because of the limited nature of template matching, the Tesseract OCR engine has an advantage for its high recognition accuracy and having the ability to retrain a character dataset [4].

ANPR tools and ways used

In [11], OpenCV was used as an image processing tool, SVM were used for contour detection on number plates and ANN were used for character recognition from number plates. The process they used had these stages: pre-processing, segmentation, morphological processing, number plate contour analysis, plate extraction and scaling, SVM, character segmentation, character contour analysis and then ANN. This mix of tools gave a 98.5% detection rate and a 90% character recognition rate.

In [12], the main tool they used was Faster R-CNN for number plate detection. They used template matching as an OCR way. The process went like this: needed database storage, input image capture, crop image to pick characters, change image to black and white, noise removal and comparing letters or numbers with template also called template matching. This got a 99.1% number plate detection accuracy.

In [14], a different way is used for ANPR which has three stages. The first is using the wavelet transform for contrast feature extraction. The reference line is then found, which is important in finding the wanted licence plate area. Lastly, licence plate adjustment is used in correctly finding the licence plate. This detection way was 92.4% right.

In [16], an edge detection algorithm was used for ANPR. It was based on things that vehicle licence plates (VLP) have like the height contrast to neighbour pixels that the contours of VLP characters have. Also, the contours of characters are closed always and the characters have a relationship. This way gave a standard measure of quality of 87.44%.

Conclusion

At image acquisition stage, the camera and OpenCV as an image processing tool worked well for most researchers. For detection, segmentation and character recognition, different tools were used with different success rates. The Pytesseract OCR engine has high recognition accuracy and can retrain a character dataset to suit different uses. The accuracy numbers of most ways were high enough to give room for trying in making an ANPR system that fits the Zimbabwean context. Some of the challenges that need to be addressed are the varying licence plate formats, the poor lighting conditions and the low quality images. Some of the possible solutions are using deep learning models, enhancing image contrast and resolution and using multiple cameras for different angles. This research shows that ANPR is a promising technology that can be applied in Zimbabwe for various purposes such as traffic management, security and law enforcement.

**CHAPTER 3**

**3.1 Introduction**

This chapter presents the results of the system analysis phase for the proposed system. The chapter consists of three main sections: information gathering tools, data analysis, and functional analysis. The first section describes the tools and techniques used to collect and organize the information about the current system and the user requirements. The second section analyses the data and models the existing system using Unified Modelling Language (UML) diagrams and data flow diagrams (DFDs). The third section defines the functional and non-functional requirements of the proposed system and illustrates them using use case diagrams. The purpose of this chapter is to provide a clear and comprehensive description of the system analysis phase and to justify the design decisions made for the proposed system.

**3.2 Description of proposed system**

Autonomous parking bay is a parking system that allows for the collection of parking fees, payment of parking fees and monitoring of parking bays without the need of human input. The current system is human intensive, it requires human input in parking duration measurement, payment, identification and reporting. This system leads to errors, miscalculations and corruption at all parking bays. The proposed system takes the human out of the equation and replaces with computer vision powered by licence plate recognition AI models as well as an integrated payment and reporting system. The robust system will collect fees hence also handle payment of parking fees through an online payment system. Furthermore, it measures duration of parking vehicles and calculate parking costs accrued. The system generates reports that allow with the management of the system at large.

3.2.1 Step-wise description of proposed system

* The driver parks in the parking bay.
* The Detection System scans the number plate using a camera and starts a timer.
* The Detection System queries the Database for the customer account associated with the number plate.
* The driver leaves the parking bay.
* The Detection System stops the timer and calculates the parking cost based on the duration.
* The Database updates the customer account with the parking cost and records the car service and fee information.
* The driver logs in to the Payment System using the number plate as an identifier.
* The Payment System retrieves the transaction history from the Database for the customer account.
* The driver pays for the parking using PayNow, a third party payment platform that is integrated with the Payment System.
* The Payment System confirms the payment and updates the payment status in the Database.
* The Reporting System retrieves the car service and fee information from the Database and generates a report of the parking activity.
* The Reporting System displays the report to a dashboard for monitoring and analysis.

**3.3 Analysis of existing system**

The current system is human intensive, it requires human input in parking duration measurement, payment, identification and reporting.

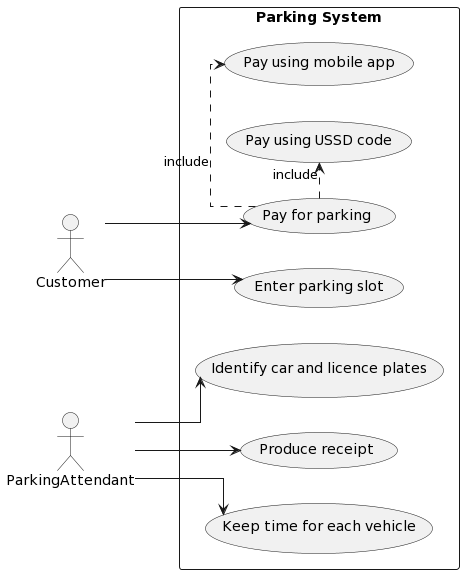
3.3.1 Stepwise description of existing system

1. Customer enters parking slot and is met by a parking attended
2. Parking attended identifies their car and licence plates
3. Customer pays for parking through parking attendant
4. Attended produces a receipt with licence plate and hours prepaid
5. Customer can also prepay for parking using mobile app
6. Customer can also prepay for parking using USSD code
7. Parking attendant will keep the time for each vehicle on the parking lot

3.3.2 UML diagrams

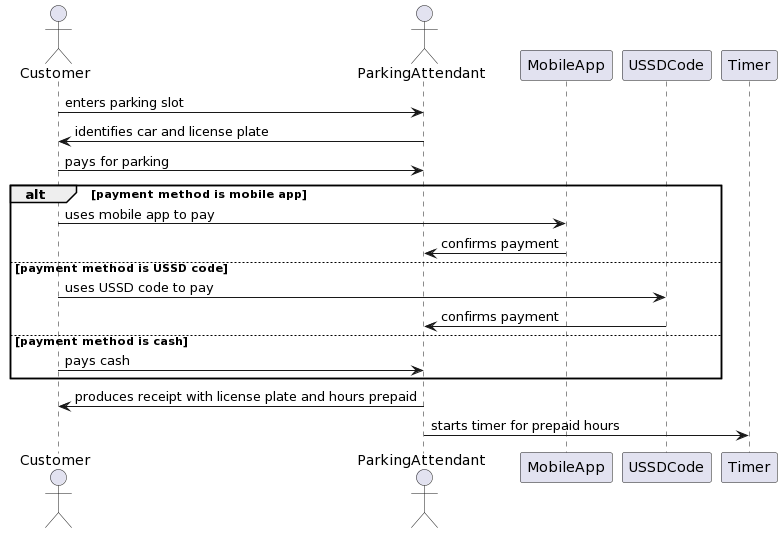
Use case

Fig 3.1 Use case diagram for Existing system

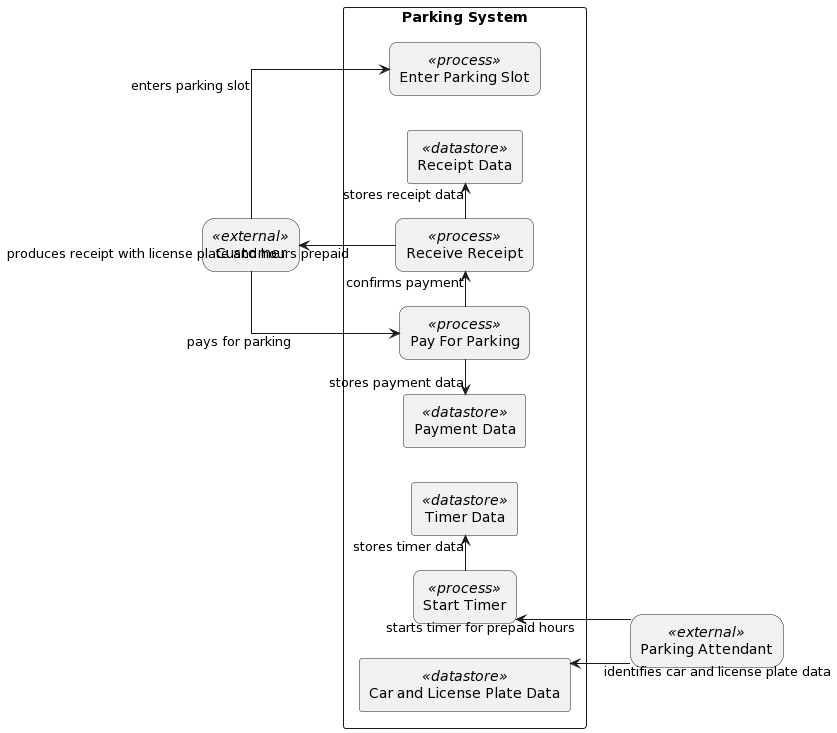


Sequence diagram

Fig 3.2 sequence diagram for Existing system



Data Flow Diagram DFD of existing system  
Fig 3.3 Data Flow diagram for existing system



3.3.3 Weaknesses of existing system

1. Humans can overestimate or underestimate hours spent on parking lot
2. Humans can be corrupt
3. Parking attendant may not be available or reliable at all times
4. Customer may lose or damage the receipt and have trouble exiting the parking slot
5. Customer may not have enough cash or mobile data to pay for parking
6. Parking system may not be scalable or efficient for large or busy parking lots
7. Humans may be dishonest or rude to the customer or the parking attendant
8. Humans may have conflicts or disputes over the parking slot or the payment

3.3.4 Functional Analysis

Functional requirements:

- The system should detect the license plate of the vehicle entering the parking lot using ANPR with open CV.

- The system should start and stop a timer for each vehicle based on the detection results.

- The system should calculate the duration and cost of parking for each vehicle based on a predefined rate.

- The system should deduct the cost from the balance of the customer's account linked to the license plate or generate an invoice for payment.

- The system should generate reports on the occupancy, revenue, and payment status of the parking lot.

- The system should allow the customer to log in to a platform using their license plate and view their transaction history and balance.

- The system should allow the customer to pay online or offline using various payment methods.

Non-functional requirements:

- The system should be accurate and reliable in detecting and recognizing license plates under different lighting and weather conditions.

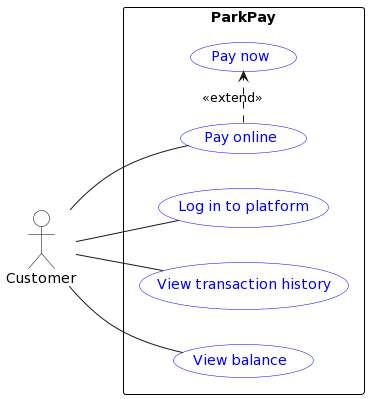
- The system should be secure and protect the privacy and data of the customers and the parking owner.

- The system should be user-friendly and provide a clear and intuitive interface for the customers and the parking owner.

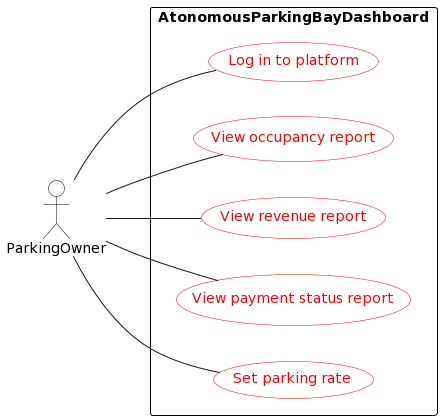
- The system should be efficient and fast in processing and transmitting data and generating feedback.

3.3.5 Use case diagrams

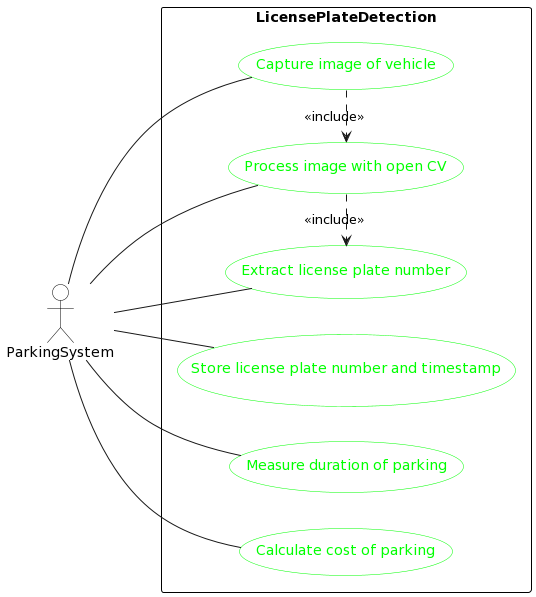
Park pay



APB Dashboard



Detection Use case



Summary

This chapter describes the process of information gathering, data analysis and functional analysis of the proposed parking system. The chapter covers the following topics:

* Information Gathering Tools, Description of System: This topic explains the tools and techniques used to collect information about the existing parking system and the user needs and expectations. The tools include interviews, questionnaires, observations and document reviews. The description of the system provides an overview of the current situation, problems and challenges faced by the parking system stakeholders.
* Data Analysis - Using UML diagrams (existing system), DFDs of existing system: This topic presents the data analysis of the existing parking system using UML diagrams and DFDs. The UML diagrams include class diagrams and sequence diagrams that model the structure and behaviour of the system components and elements. The DFDs include context diagrams, level 0 diagrams and level 1 diagrams that model the data flow and processes in the system.
* Functional Analysis of Proposed System – Functional and Non-Functional Requirements, User Case Diagrams: This topic presents the functional analysis of the proposed parking system using functional and non-functional requirements and user case diagrams. The functional requirements specify what the system should do and provide in terms of features and functions. The non-functional requirements specify how the system should perform and behave in terms of quality attributes such as usability, reliability, security, etc. The user case diagrams model the interactions and scenarios between the system and its users.
* The chapter demonstrates that the proposed parking system is designed according to the information gathered, data analysed and functional requirements specified. The chapter also provides valuable insights and recommendations for future design and development of the system.

**CHAPTER 4**

**Introduction**

This chapter presents the design of the parking system, which is a software solution that provides parking services to customers and stakeholders. The design of the system covers various aspects, such as systems diagrams, architectural design, database design, program design, and interface design. Each aspect describes the structure, behaviour, and interaction of the system components and elements. The design of the system is based on the system specification document and the user acceptance criteria defined in the previous chapter. The chapter covers the following topics:

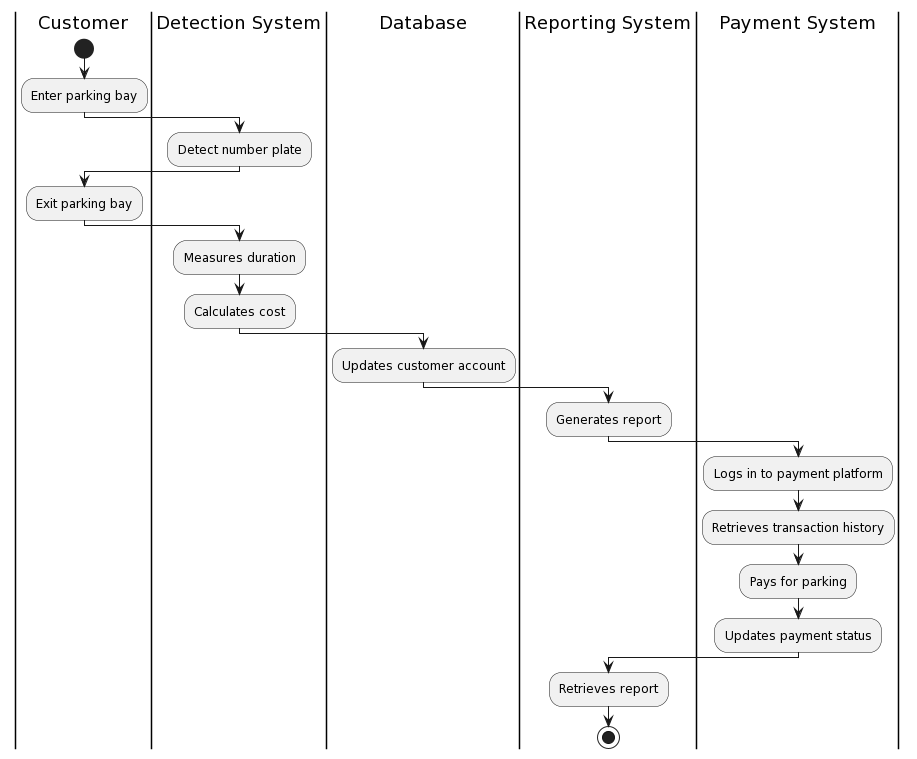
* **Systems Diagrams** – Using UML diagrams, DFD diagrams, Activity diagrams to represent the static and dynamic aspects of the system.
* **Architectural Design** – Describing the hardware and networking components of the system and how they interact.
* **Database Design** – Creating ER diagrams and Normalized Databases to store and manage the data of the system.
* **Program Design** – Using Class diagrams, Sequence diagrams, Package diagrams, Pseudocode to define the classes, methods, and interactions of the system.
* **Interface Design** – Designing the user interface of the system using screenshots.

The chapter aims to provide an overview of software modeling and design techniques and tools that can help in developing high-quality software systems

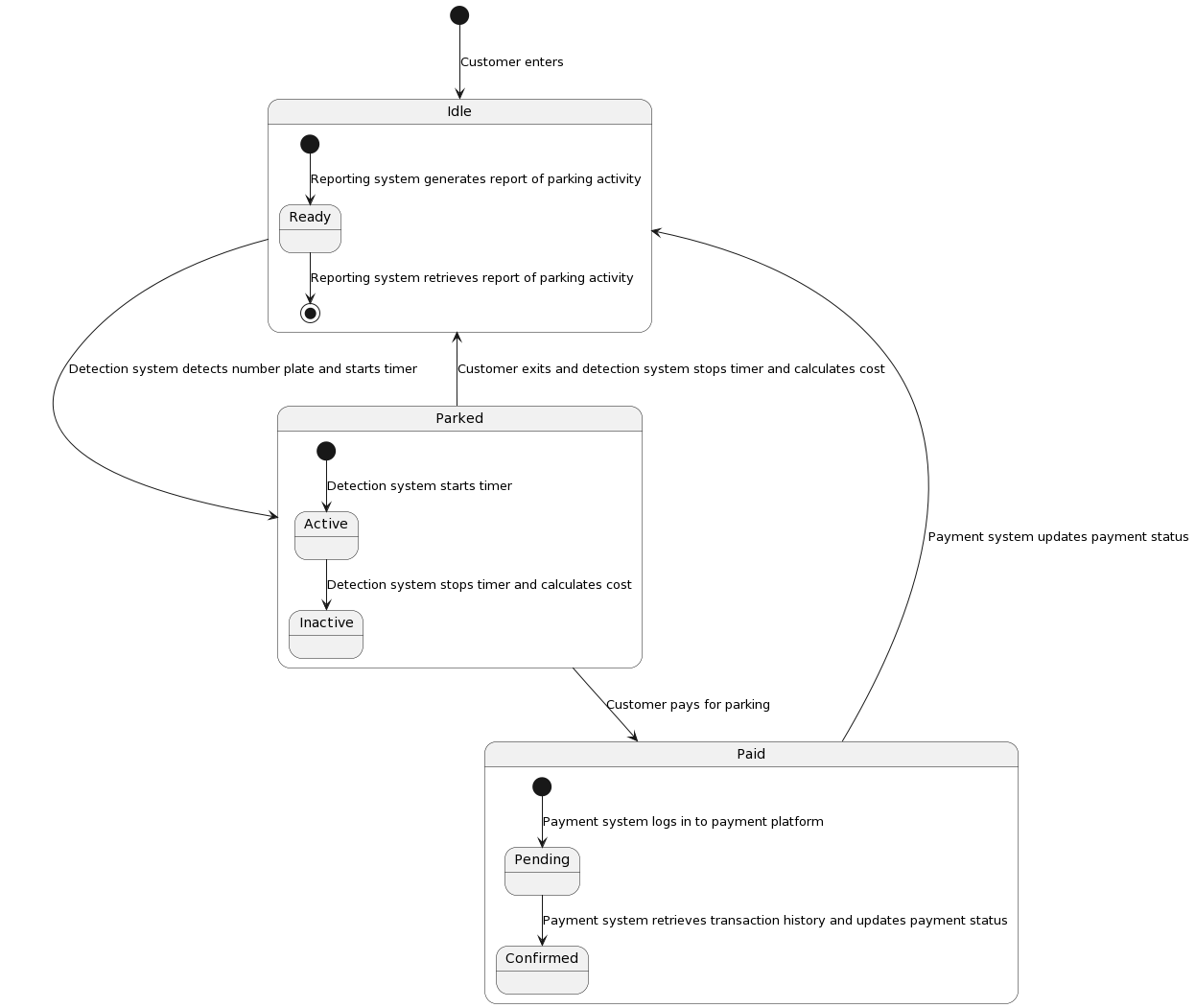
**4.1 Systems Diagrams**

Systems diagrams are graphical representations that show the main components and elements of the system and their relationships. Systems diagrams help to visualize and understand the system as a whole and its parts. There are different types of systems diagrams that can be used to model different aspects of the system, such as UML diagrams, DFD diagrams, and activity diagrams. UML diagrams are a standard notation for modeling object-oriented systems. UML diagrams include class diagrams, sequence diagrams, package diagrams, etc. DFD diagrams are a notation for modeling data flows and processes in a system. DFD diagrams include context diagrams, level 0 diagrams, level 1 diagrams and level 2. Activity diagrams are a notation for modeling the dynamic behavior and flow of control in a system.

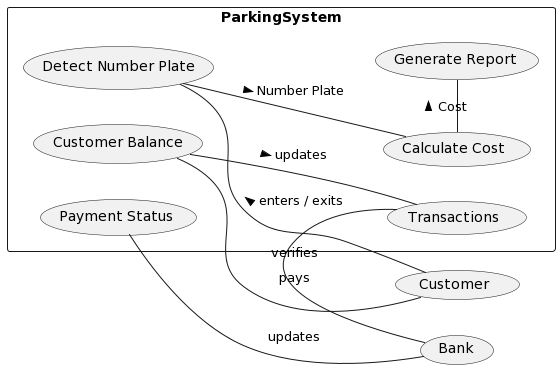
Activity diagram

****

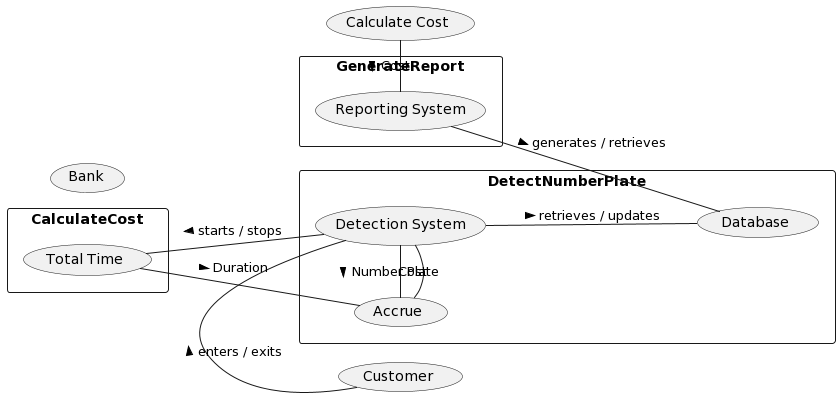
State diagrams



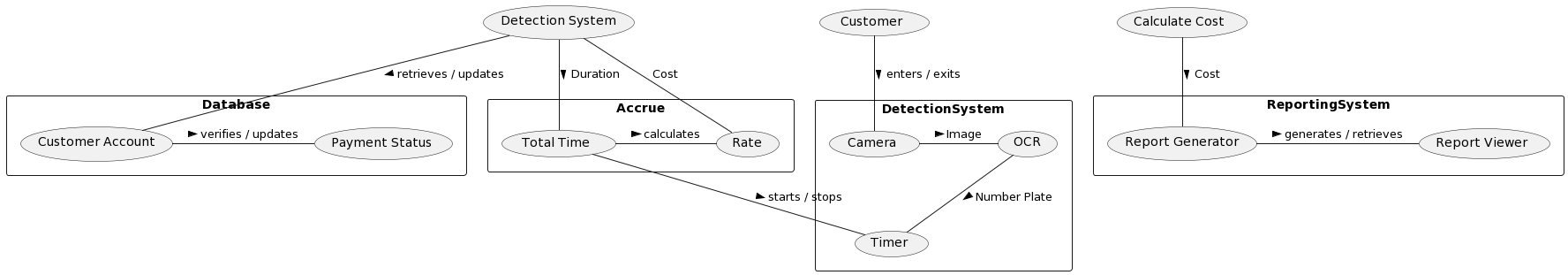
Data flow diagrams   
  
context



DFD level 1



DFD level 2



**4.2 Architectural design**

Architectural design is the process of defining the high-level structure and organization of the system components and their interactions. Architectural design helps to determine how the system will meet the functional and non-functional requirements and how it will be deployed and maintained. Architectural design includes hardware design and networking design. Hardware design is the process of selecting and configuring the physical devices and equipment that support the system functionality and performance. Hardware design includes servers, computers, cameras, sensors, etc. Networking design is the process of designing and implementing the communication infrastructure that connects the system components and enables data exchange.

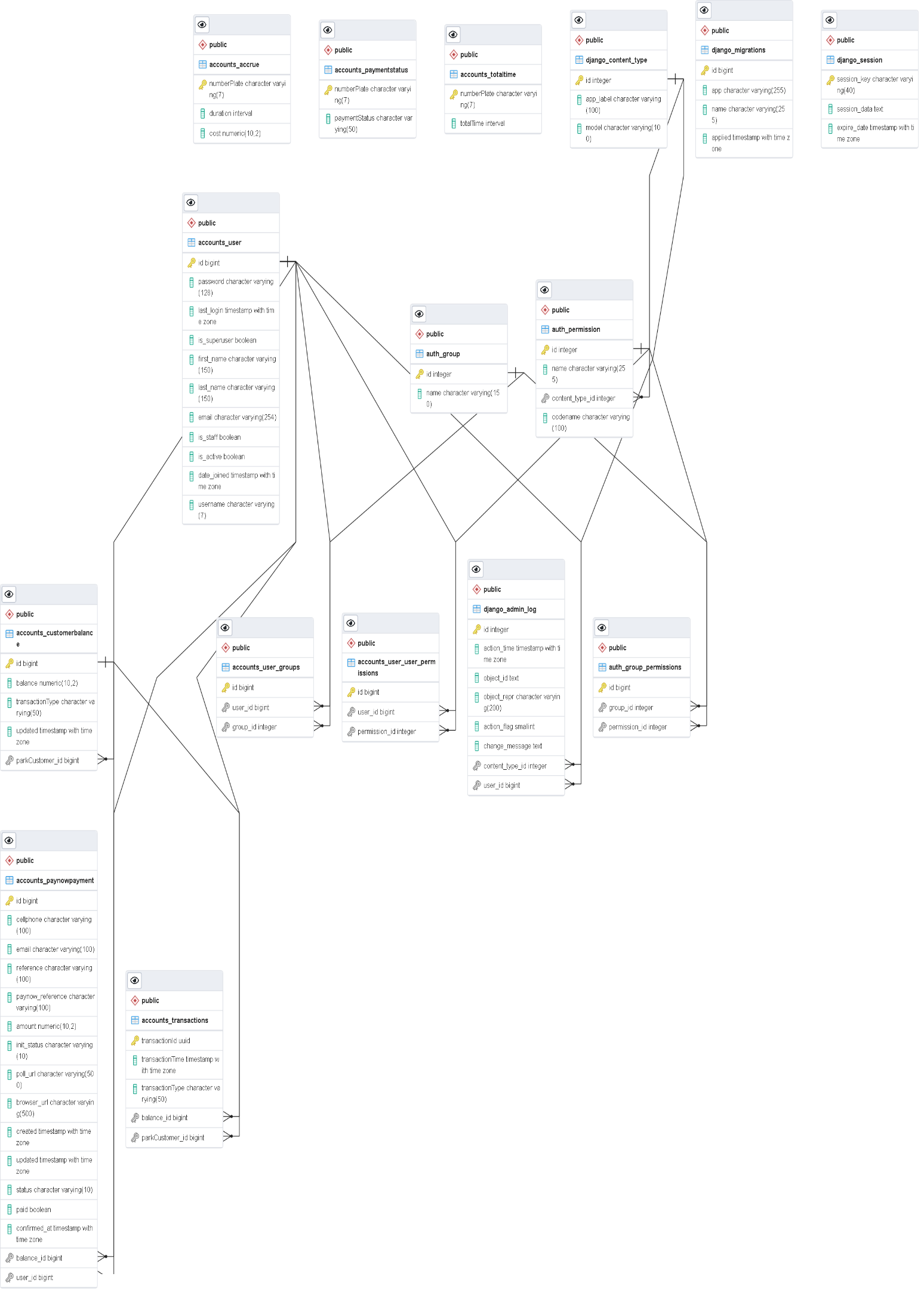


The diagram shows the networking between the hardware components of the system. It shows the three systems the detection system, the reporting system and park pay are connected to a universal database server. The detection systems has access to a bus network of cameras located at various parking bays. It shows how customers can access park pay from their devices remotely.

**4.3 Database design**

Database design is the process of designing and implementing the logical structure and organization of the data that is stored and manipulated by the system. Database design helps to ensure data integrity, security, consistency, and efficiency. Database design includes ER diagrams and normalized databases. ER diagrams are a notation for modeling the entities, attributes, and relationships in a database. ER diagrams include entity sets, relationship sets, primary keys, foreign keys, etc. Normalized databases are databases that follow certain rules or normal forms to reduce data redundancy and anomalies. Normalized databases include first normal form (1NF), second normal form (2NF), third normal form (3NF), etc.

**ER Diagram**



Normalized database

TABLE 4.1

1NF

User

|  |  |
| --- | --- |
| **username** | **…** |
| PK |  |

TABLE 4.2

Customerbalance

|  |  |  |  |
| --- | --- | --- | --- |
| **parkCustomer** | **balance** | **transactionType** | **updated** |
| PK FK |  |  |  |

TABLE 4.3

Transactions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **parkCustomer** | **transactionId** | **transactionTime** | **transactionType** | **balance** |
| PK FK | PK |  |  | FK |

TABLE 4.4

PaymentStatus

|  |  |
| --- | --- |
| **numberPlate** | **paymentStatus** |
| PK |  |

TABLE 4.5

PaynowPayment

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **user** | **cellphone** | **email** | **reference** | **paynow\_reference** | **amount** | **init\_status** | **poll\_url** | **browser\_url** | **created** | **updated** | **status** | **paid** | **confirmed\_at** | **balance** |
| PK FK |  |  |  |  |  |  |  |  |  |  |  |  |  | FK |

TABLE 4.6

Accrue

|  |
| --- |
| **numberPlate** |
| PK |

TABLE 4.7

TotalTime

|  |
| --- |
| **numberPlate** |
| PK |

2NF

TABLE 4.8  
User

|  |  |
| --- | --- |
| **username** | **…** |
| PK |  |

TABLE 4.9

Customerbalance

|  |  |  |  |
| --- | --- | --- | --- |
| **parkCustomer** | **balance** | **transactionType** | **updated** |
| PK FK |  |  |  |

TABLE 4.10  
Transactions

|  |  |  |  |
| --- | --- | --- | --- |
| **parkCustomer** | **transactionId** | **transactionTime** | **transactionType** |
| PK FK | PK |  |  |

TABLE 4.11  
PaymentStatus

|  |  |
| --- | --- |
| **numberPlate** | **paymentStatus** |
| PK |  |

TABLE 4.12  
PaynowPayment

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **user** | **cellphone** | **email** | **reference** | **paynow\_reference** | **amount** | **init\_status** | **poll\_url** | **browser\_url** | **created** | **updated** | **status** | **paid** | **confirmed\_at** |
| PK FK |  |  |  |  |  |  |  |  |  |  |  |  |  |

TABLE 4.13  
Accrue

|  |
| --- |
| **numberPlate** |
| PK |

TABLE 4.14  
TotalTime

|  |
| --- |
| **numberPlate** |
| PK |

TABLE 4.15  
CustomerBalanceHistory

|  |  |
| --- | --- |
| parkCustomer | balance |
| PK  FK |  |

3NF

TABLE 4.16  
User

|  |  |
| --- | --- |
| **username** | **…** |
| PK |  |

TABLE 4.17  
Customerbalance

|  |  |  |  |
| --- | --- | --- | --- |
| **parkCustomer** | **balance** | **transactionType** | **updated** |
| PK FK |  |  |  |

TABLE 4.18  
Transactions

|  |  |  |  |
| --- | --- | --- | --- |
| **parkCustomer** | **transactionId** | **transactionTime** | **transactionType** |
| PK FK | PK |  |  |

TABLE 4.19  
PaymentStatus

|  |  |
| --- | --- |
| **numberPlate** | **paymentStatus** |
| PK |  |

TABLE 4.20  
PaynowPayment

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **user** | **cellphone** | **email** | **reference** | **paynow\_reference** | **amount** | **init\_status** | **poll\_url** | **browser\_url** | **created** | **updated** | **paid** | **confirmed\_at** |
| PK FK |  |  |  |  |  |  |  |  |  |  |  |  |

TABLE 4.21  
Accrue

|  |
| --- |
| **numberPlate** |
| PK |

TABLE 4.22  
TotalTime

|  |
| --- |
| **numberPlate** |
| PK |

TABLE 4.23  
CustomerBalanceHistory

|  |  |
| --- | --- |
| parkCustomer | balance |
| PK  FK |  |

TABLE 4.24

CostHistory

|  |  |
| --- | --- |
| numberPlate | cost |
| PK |  |

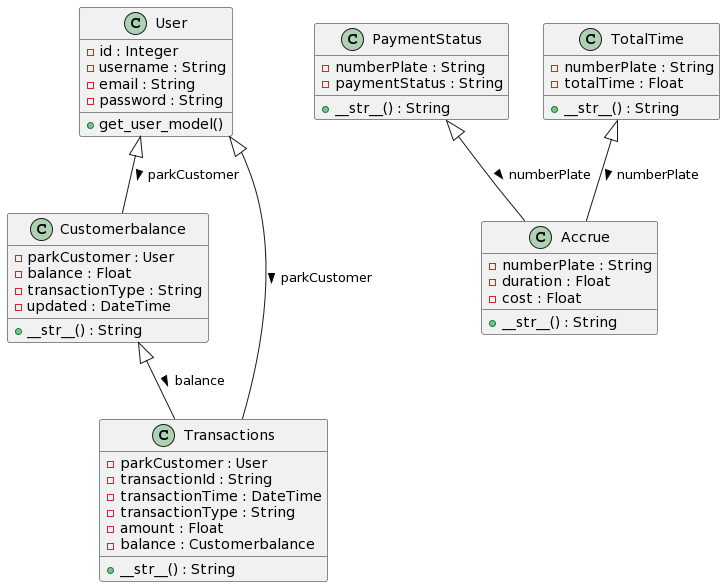
TABLE 4.25  
PaymentStatusHistory

|  |  |
| --- | --- |
| user | status |
| PK  FK |  |

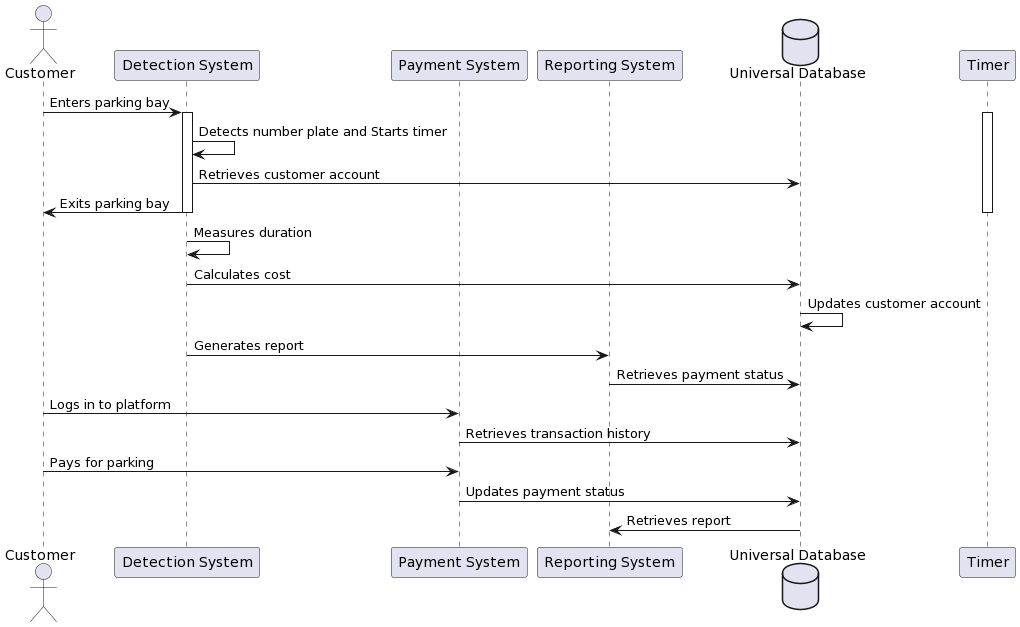
**4.4 Program design**

Program design is the process of designing and implementing the software modules and components that provide the system functionality and behaviour. Program design helps to ensure code quality, readability, maintainability, and reusability. Program design includes class diagrams, sequence diagrams, package diagrams, pseudocode. Class diagrams are a notation for modelling the classes, attributes, methods, and associations in an object-oriented program. Class diagrams include classes, inheritance relationships, aggregation relationships, composition relationships, etc. Sequence diagrams are a notation for modelling the interactions and messages between objects in a program. Sequence diagrams include objects, lifelines, messages, activation bars, etc. Package diagrams are a notation for modelling the organization and dependencies of packages in a program. Package diagrams include packages, classes and dependencies.

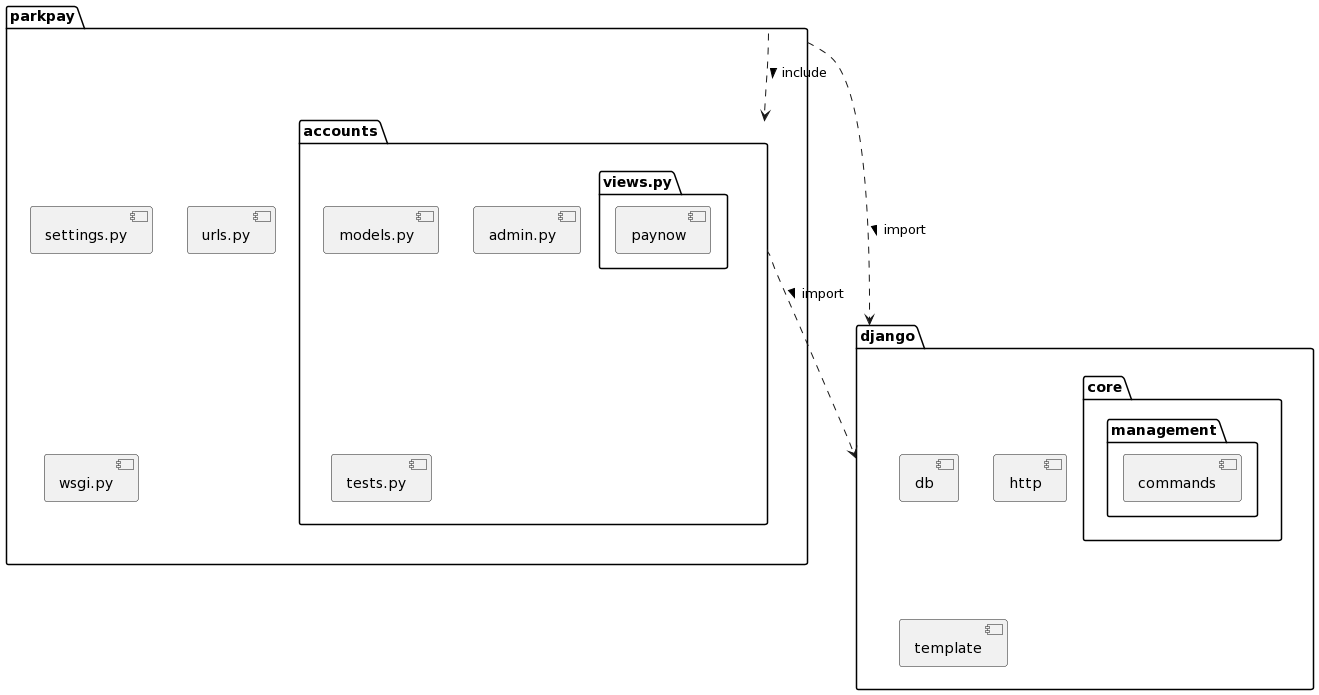
Class diagram



Sequence diagram



Package diagram



**4.5 Interface design**

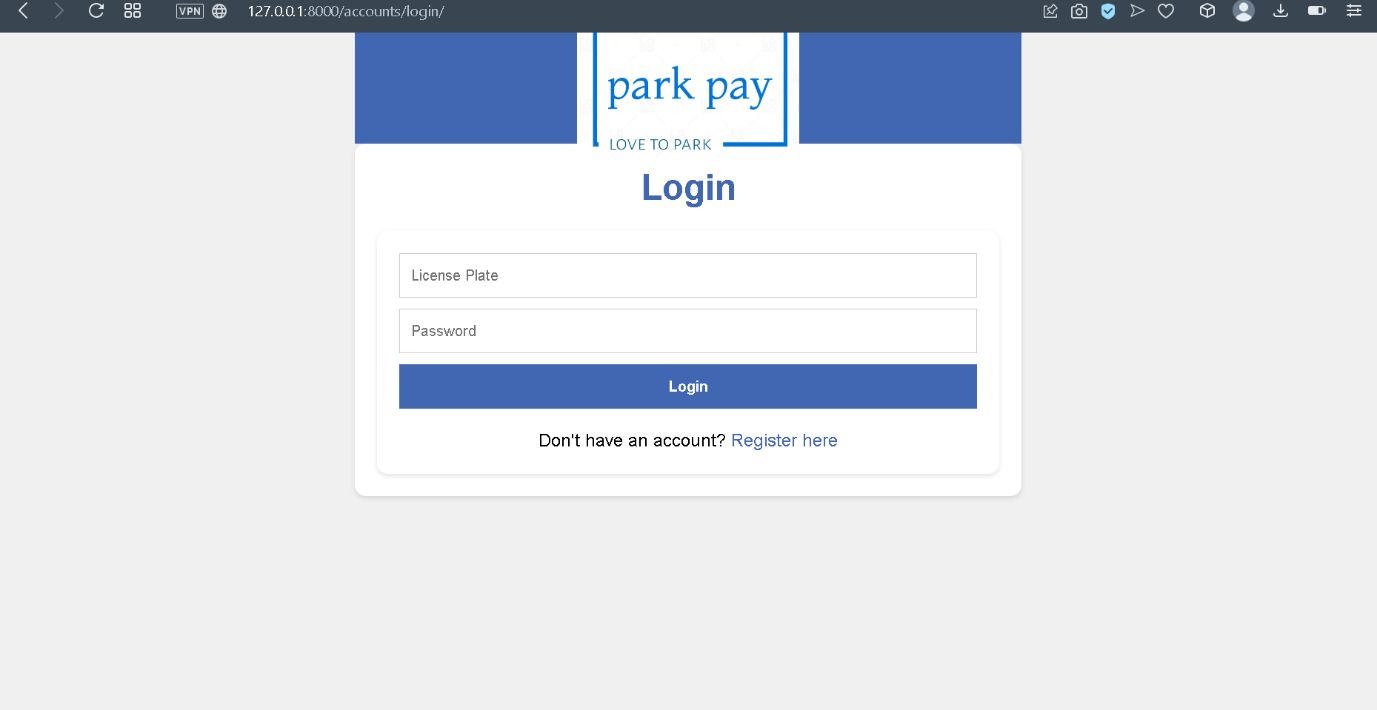
Interface Design Introduction:

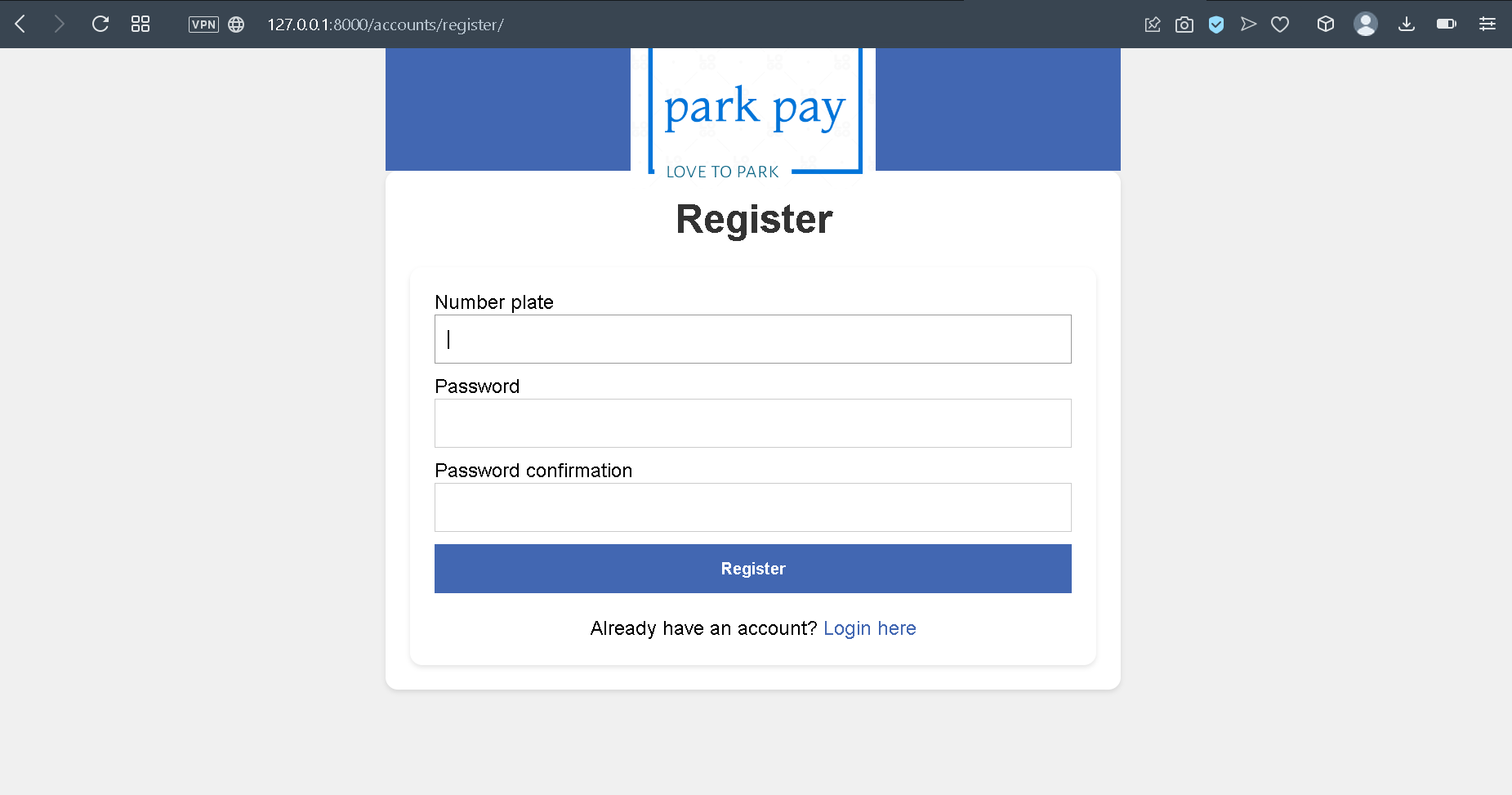
Interface design is the process of designing and implementing the user interface of the system. The user interface is the part of the system that interacts with the users and allows them to access the system functionality and information. Interface design helps to ensure user satisfaction, usability, accessibility, and aesthetics. Interface design includes screenshots of user interface. Screenshots are images that capture the appearance and layout of user interface elements, such as windows, menus, buttons, text boxes and icons.

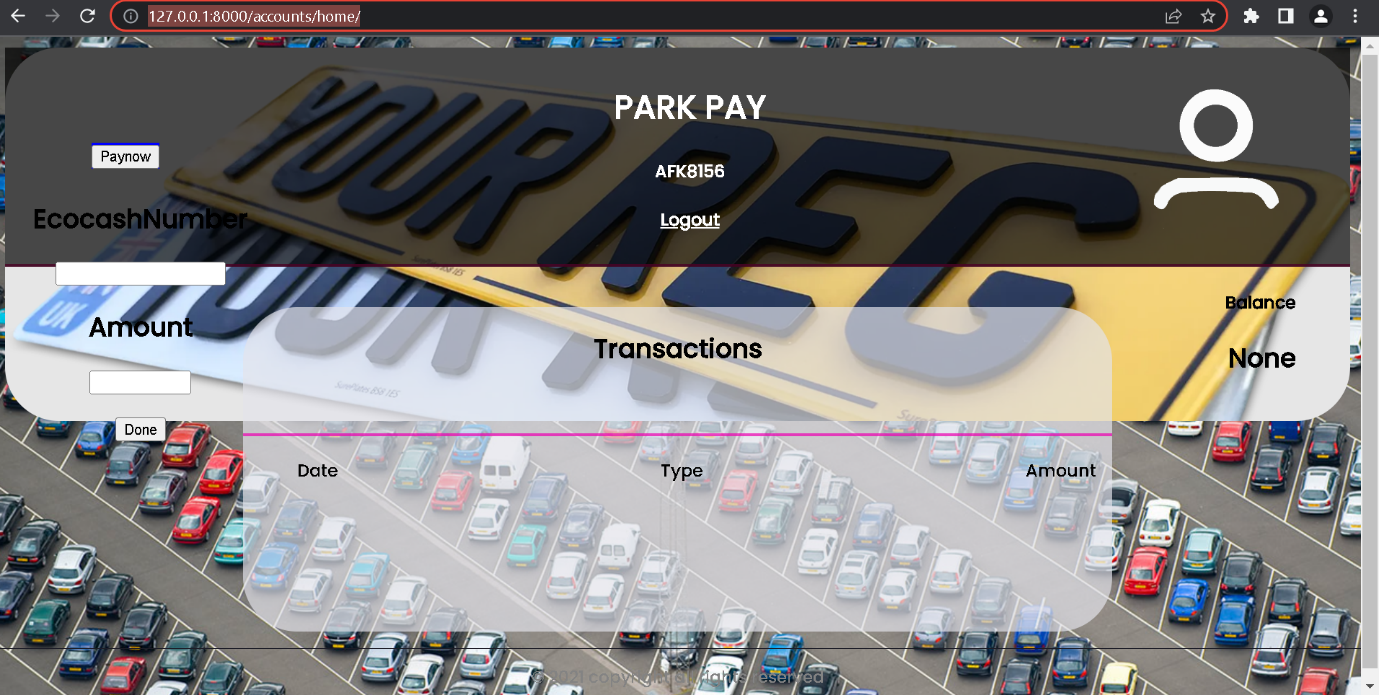
The chapter demonstrated that the parking system was designed according to the specified requirements and acceptance criteria. The chapter also provided valuable insights and recommendations for future enhancements and maintenance of the system.

Draw the interface, not the actual one at this stage

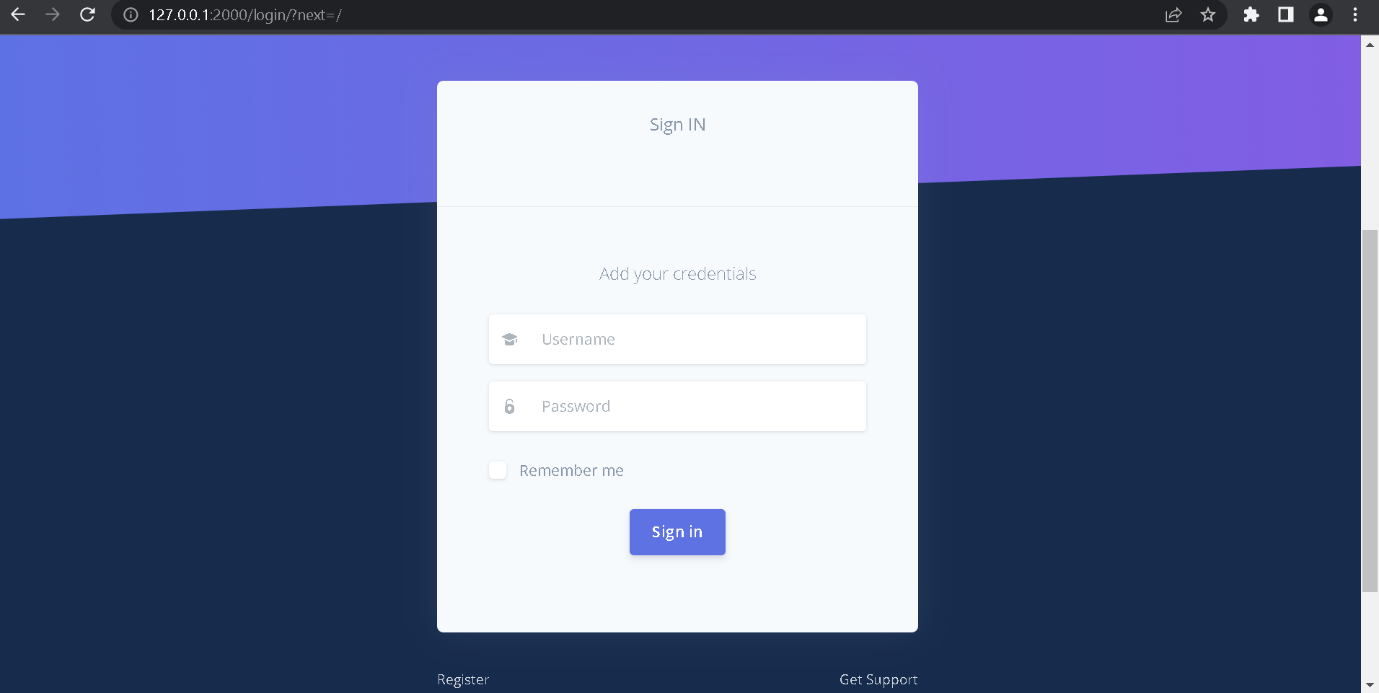
PARK PAY

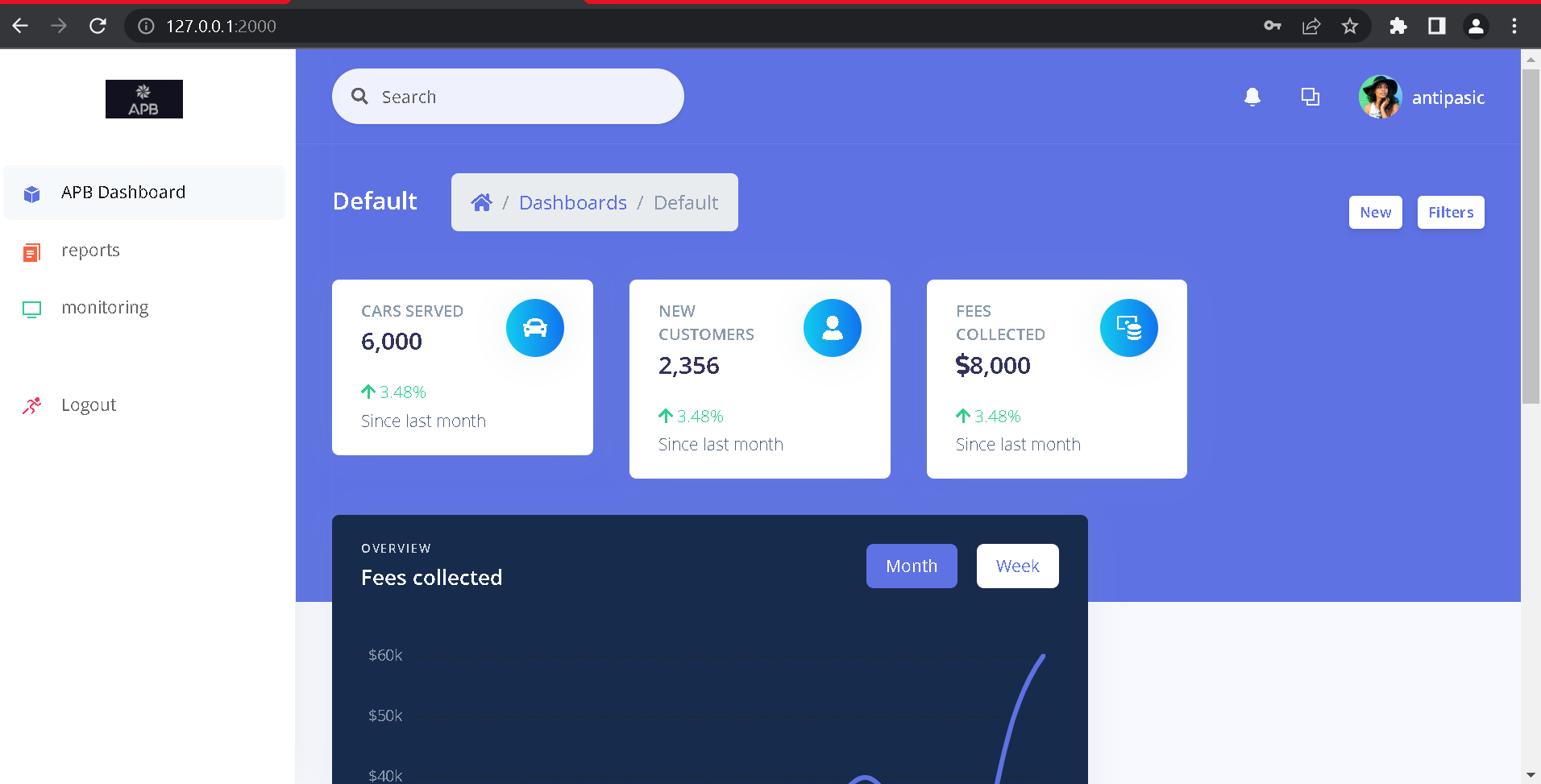






Reporting system - APB Dashboard

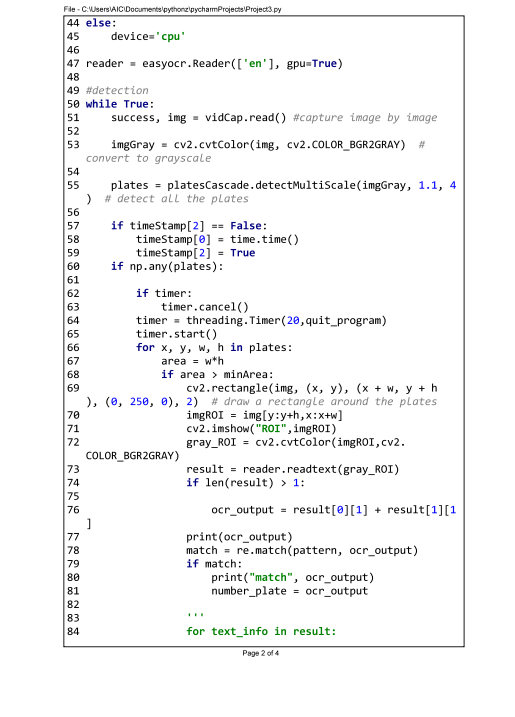
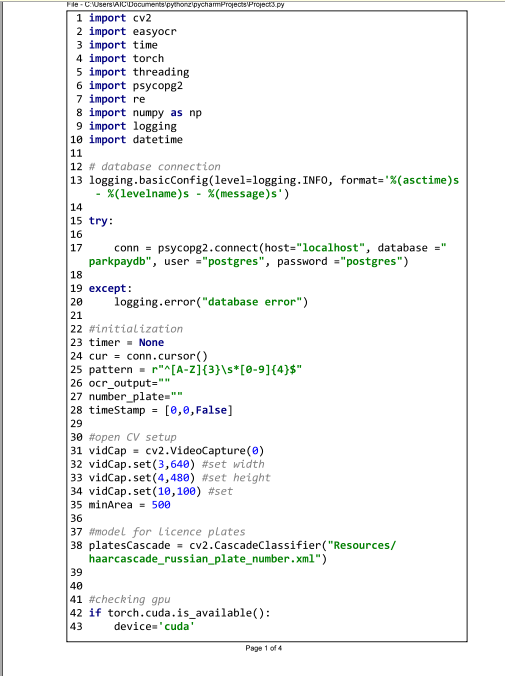


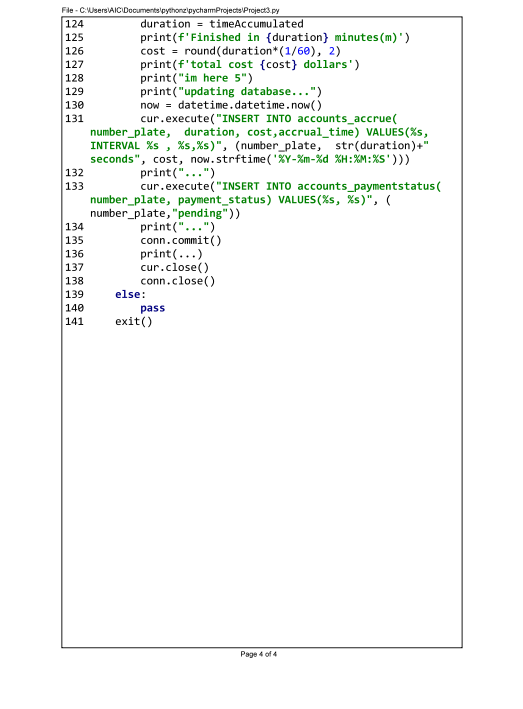
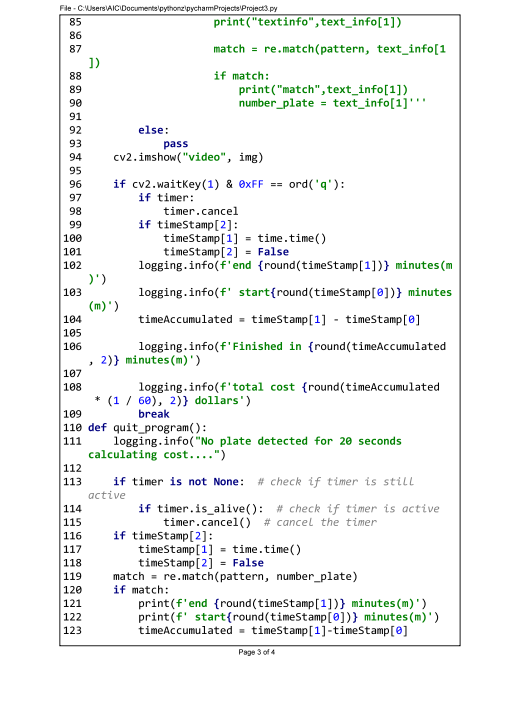


**CHAPTER 5**

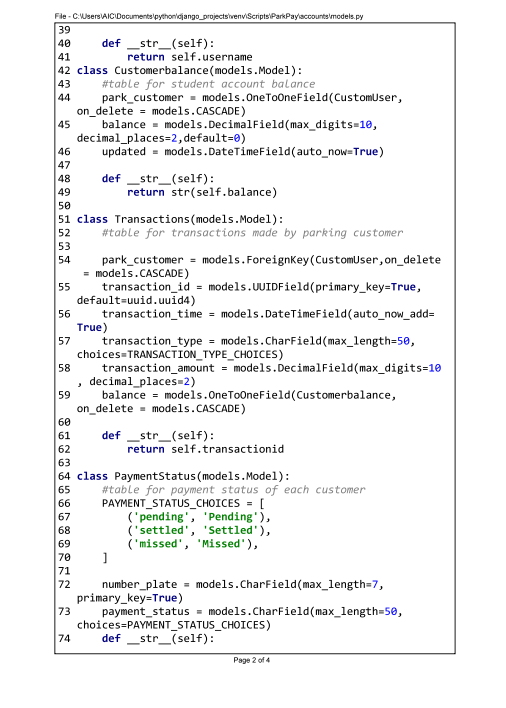
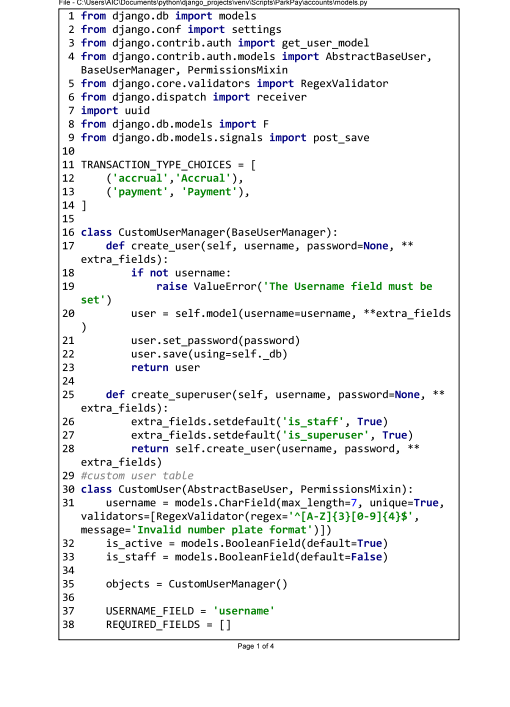
**5.1 Sample code**

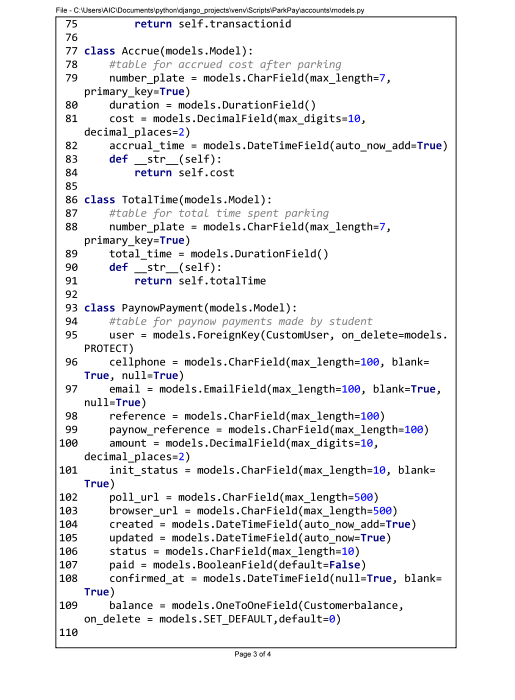
Detection system





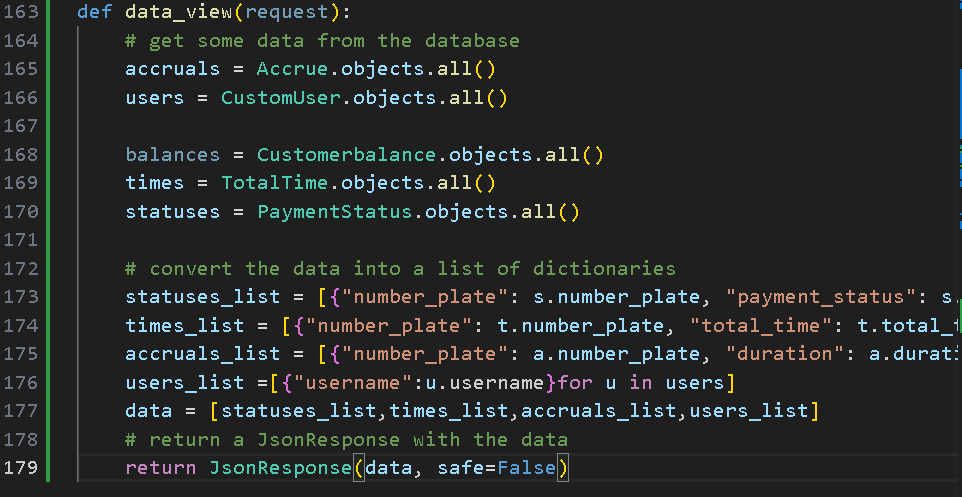
System Models

****

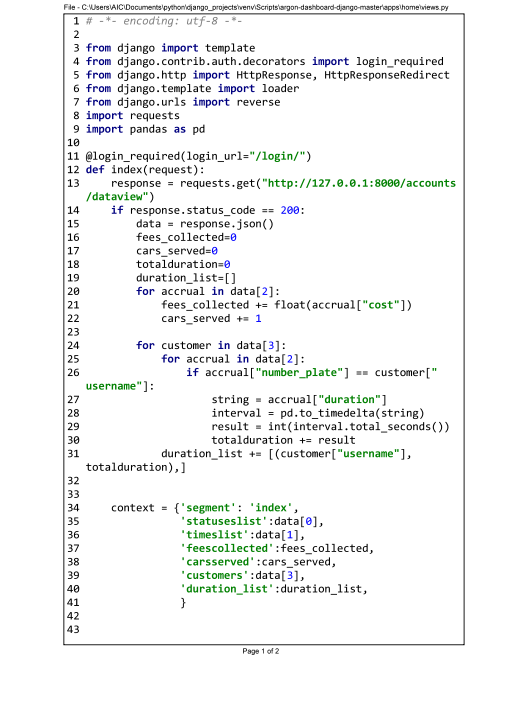
****

ParkPay – Reporting system API

Parkpay views.py

****

APB dashboard views.py



**5.2.1 Unit testing**

TABLE 5.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Component | Function | Input | Expected Output | Actual Output | Pass/Fail |
| Detection System | scan\_number\_plate | image of a car with a number plate | number plate as a string | number plate as a string | Pass |
| Detection System | start\_timer | none | current time as a timestamp | current time as a timestamp | Pass |
| Detection System | stop\_timer | none | current time as a timestamp and duration as a number | current time as a timestamp and duration as a number | Pass |
| Detection System | calculate\_cost | duration as a number and rate as a number | cost as a number | cost as a number | Pass |
| Detection System | query\_database | number plate as a string | customer account as an object or null if not found | customer account as an object or null if not found | Pass |
| Payment System | login\_user | number plate as a string and password as a string or token as a string | user session as an object or error message if invalid credentials or token | user session as an object or error message if invalid credentials or token | Pass |
| Payment System | retrieve\_transaction\_history | user session as an object and customer account as an object or null if not found | transaction history as an array of objects or error message if invalid session or account | transaction history as an array of objects or error message if invalid session or account | Pass |
| Payment System | pay\_using\_paynow | user session as an object, customer account as an object, cost as a number and payment details as an object | payment confirmation as an object or error message if invalid session, account, cost or payment details | payment confirmation as an object or error message if invalid session, account, cost or payment details | Pass |
| Payment System | update\_payment\_status | user session as an object, customer account as an object and payment confirmation as an object | payment status update in the Database and success message or error message if invalid session, account or confirmation | payment status update in the Database and success message or error message if invalid session, account or confirmation | Pass |
| Reporting system | retrieve\_car\_service\_and\_fee\_information | none | car service and fee information as an array of objects from the Database or error message if Database is not accessible | car service and fee information as an array of objects from the Database or error message if Database is not accessible | Pass |
| Reporting system | generate\_report | car service and fee information as an array of objects | report of the parking activity as a file or error message if car service and fee information is invalid or empty | report of the parking activity as a file or error message if car service and fee information is invalid or empty | Pass |
| Reporting system | display\_report | report of the parking activity as a file | report displayed to a dashboard or error message if report file is invalid or empty | report displayed to a dashboard or error message if report file is invalid or empty | Pass |

**5.2.2 Module testing**

* Module Testing for the Payment System:

TABLE 5.2

|  |  |
| --- | --- |
| Module Name | Test Case Description |
| login\_user | Test if the function can log in the user to the Payment System using the number plate and password or token. Check if the output is a user session object that contains the user information and session ID or an error message if the credentials or token are invalid. |
| retrieve\_transaction\_history | Test if the function can retrieve the transaction history from the Database for the customer account using the user session object. Check if the output is an array of transaction objects that contain the date, time, amount and status of each transaction or an error message if the user session or customer account are invalid. |
| pay\_using\_paynow | Test if the function can pay for the parking using PayNow, a third party payment platform that is integrated with the Payment System. Check if the output is a payment confirmation object that contains the transaction ID, amount and status or an error message if the user session, customer account, cost or payment details are invalid. |
| update\_payment\_status | Test if the function can update the payment status in the Database using the user session, customer account and payment confirmation objects. Check if the output is a success message that indicates that the payment status has been updated or an error message if the user session, customer account or payment confirmation are invalid. |

* Module Testing for the Reporting System:

|  |  |
| --- | --- |
| Module Name | Test Case Description |
| retrieve\_car\_service\_and\_fee\_information | Test if the function can retrieve the car service and fee information from the Database. Check if the output is an array of car service and fee objects that contain the number plate, duration and cost of each car service or an error message if the Database is not accessible. |
| generate\_report | Test if the function can generate a report of the parking activity using the car service and fee information. Check if the output is a report file that contains a summary and a table of the car service and fee information or an error message if the car service and fee information is invalid or empty. |
| display\_report | Test if the function can display the report to a dashboard for monitoring and analysis. Check if the output is a dashboard that shows a graphical representation of the report data or an error message if the report file is invalid or empty. |

* Module Testing for the Database:

|  |  |
| --- | --- |
| Module Name | Test Case Description |
| update\_customer\_account | Test if the function can update the customer account with the cost of parking using the number plate and the cost as inputs. Check if the output is a success message that indicates that the customer account has been updated or an error message if the number plate or the cost are invalid. |
| record\_car\_service\_and\_fee\_information | Test if the function can record the car service and fee information using the number plate, duration and cost as inputs. Check if the output is a success message that indicates that the car service and fee information has been recorded or an error message if the number plate, duration or cost are invalid. |
| update\_payment\_status | Test if the function can update the payment status using the number plate and the payment confirmation as inputs. Check if the output is a success message that indicates that the payment status has been updated or an error message if the number plate or the payment confirmation are invalid. |
| retrieve\_transaction\_history | Test if the function can retrieve the transaction history using the number plate as an input. Check if the output is an array of transaction objects that contain the transaction ID, time, type and balance of each transaction or an error message if the number plate is invalid. |
| retrieve\_car\_service\_and\_fee\_information | Test if the function can retrieve the car service and fee information. Check if the output is an array of car service and fee objects that contain the number plate, duration and cost of each car service. |

**5.2.3 Integration Testing**

TABLE 5.3

|  |  |  |
| --- | --- | --- |
| Component 1 | Component 2 | Test Case Description |
| Detection System | Database | Test if the Detection System can query the Database for the customer account using the number plate and update the customer account with the cost of parking using the duration and rate. Check if the Database returns the customer account or null if not found and updates the customer account with the cost or returns an error message if invalid. |
| Payment System | Database | Test if the Payment System can retrieve the transaction history from the Database for the customer account using the user session and update the payment status in the Database using the user session, customer account and payment confirmation. Check if the Database returns the transaction history or an error message if invalid and updates the payment status or returns an error message if invalid. |
| Reporting System | Database | Test if the Reporting System can retrieve the car service and fee information from the Database and generate a report of the parking activity using the car service and fee information. Check if the Database returns the car service and fee information or an error message if not accessible and the Reporting System generates a report file or an error message if invalid or empty. |

**5.2.4 System testing**

System Testing: This level tests the functionality and performance of the whole system.

|  |
| --- |
| **Test Case Description** |
| 1. Test if the system can detect and read the number plate of a car that parks in the parking bay and starts a timer. |
| 1. Test if the system can calculate the cost of parking based on the duration and rate and update the customer account with the cost. |
| 1. Test if the system can log in the user to the Payment System using the number plate and password or token and retrieve the transaction history for the customer account. |
| 1. Test if the system can pay for the parking using PayNow, a third party payment platform that is integrated with the Payment System and update the payment status in the Database. |
| 1. Test if the system can generate a report of the parking activity using the car service and fee information from the Database and display it to a dashboard for monitoring and analysis. |
| 1. Test if the system can handle multiple cars parking and leaving at different times and rates. |
| 1. Test if the system can handle invalid inputs such as invalid number plates, passwords, tokens, payment details, etc. |
| 1. Test if the system can handle errors such as Database not accessible, Payment System not available, PayNow not working, etc. |
| 1. Test if the system can meet the performance requirements such as response time, accuracy, reliability, scalability, security, etc. |

**5.2.5 Acceptance testing**

Acceptance Testing: This level tests if the system meets the user expectations and requirements.

|  |
| --- |
| **Test Case Description** |
| 1. Test if the user can park and leave the parking bay without any hassle or delay. |
| 1. Test if the user can view their customer account and transaction history on the Payment System. |
| 1. Test if the user can pay for their parking using their preferred payment method on PayNow. |
| 1. Test if the user can receive a report of their parking activity via email. |
| 1. Test if the user is satisfied with the quality and usability of the system. |

**5.3.1 Installation and Deployment:**

* Install the required software and hardware for the system, such as Python, Django, cv2, easyocr, torch, PostgreSQL, cameras, etc.
* Configure the system settings, such as the rate of parking, the password or token for the Payment System, the email address for the Reporting System, etc.
* Run the system code and test if it works as expected.
* Deploy the system to a server or a cloud platform that can handle the traffic and storage needs of the system.
* Monitor the system performance and functionality and fix any bugs or errors that may occur.

**5.3.2 Maintenance:**

* Update the system software and hardware regularly to ensure security and compatibility.
* Backup the system data regularly to prevent data loss or corruption.
* Review the system requirements and feedback from the users and make any changes or improvements as needed.
* Document the system design, code, testing and deployment processes and keep them updated.

**CHAPTER 6**

**6.1 Results**

TABLE 6.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Component | Function | Input | Expected Output | Actual Output | Pass/Fail |
| Detection System | scan\_number\_plate | image of a car with a number plate | number plate as a string | number plate as a string | Pass |
| Detection System | start\_timer | none | current time as a timestamp | current time as a timestamp | Pass |
| Detection System | stop\_timer | none | current time as a timestamp and duration as a number | current time as a timestamp and duration as a number | Pass |
| Detection System | calculate\_cost | duration as a number and rate as a number | cost as a number | cost as a number | Pass |
| Detection System | query\_database | number plate as a string | customer account as an object or null if not found | customer account as an object or null if not found | Pass |
| Payment System | login\_user | number plate as a string and password as a string or token as a string | user session as an object or error message if invalid credentials or token | user session as an object or error message if invalid credentials or token | Pass |
| Payment System | retrieve\_transaction\_history | user session as an object and customer account as an object or null if not found | transaction history as an array of objects or error message if invalid session or account | transaction history as an array of objects or error message if invalid session or account | Pass |
| Payment System | pay\_using\_paynow | user session as an object, customer account as an object, cost as a number and payment details as an object | payment confirmation as an object or error message if invalid session, account, cost or payment details | payment confirmation as an object or error message if invalid session, account, cost or payment details |  |
| Payment System | update\_payment\_status | user session as an object, customer account as an object and payment confirmation as an object | payment status update in the Database and success message or error message if invalid session, account or confirmation | payment status update in the Database and success message or error message if invalid session, account or confirmation | Pass |
| Reporting System | retrieve\_car\_service\_and\_fee\_information | none | car service and fee information as an array of objects from the Database or error message if Database is not accessible | car service and fee information as an array of objects from the Database or error message if Database is not accessible | Pass |
| Reporting System | generate\_report | car service and fee information as an array of objects | report of the parking activity as a file or error message if car service and fee information is invalid or empty | report of the parking activity as a file or error message if car service and fee information is invalid or empty | Pass |
| Reporting System | display\_report | report of the parking activity as a file | report displayed to a dashboard or error message if report file is invalid or empty | report displayed to a dashboard or error message if report file is invalid or empty | Pass |
| Database | update\_customer\_account | customer account as an object and cost as a number | customer account updated with the cost and success message or error message if customer account or cost is invalid | customer account updated with the cost and success message or error message if customer account or cost is invalid | Pass |
| Database | record\_car\_service\_and\_fee\_information | number plate as a string, duration as a number and cost as a number | car service and fee information recorded and success message or error message if number plate, duration or cost is invalid | car service and fee information recorded and success message or error message if number plate, duration or cost is invalid | Pass |
| Database | update\_payment\_status | customer account as an object and payment confirmation as an object | payment status updated and success message or error message if customer account or payment confirmation is invalid | payment status updated and success message or error message if customer account or payment confirmation is invalid | Pass |
| Database | retrieve\_transaction\_history | customer account as an object | transaction history as an array of objects or error message if customer account is invalid | transaction history as an array of objects or error message if customer account is invalid | Pass |
| Database | retrieve\_car\_service\_and\_fee\_information | none | car service and fee information as an array of objects | car service and fee information as an array of objects | Pass |

6.1.2 Integration Testing Results:

* The purpose of integration testing was to verify the functionality and performance of groups of modules or units that are integrated together in the parking system.
* The scope of integration testing covered four main groups: Detection System-Database group, Payment System-Database group, Reporting System-Database group, and Detection System-Payment System group.
* The focus area of integration testing was to ensure that the interfaces and interactions between modules were correct and consistent.
* The test cases for integration testing were designed based on the use cases and scenarios defined in the system specification document.
* The test cases for integration testing were executed using a combination of manual and automated methods.
* The test cases for integration testing were evaluated based on the expected outcomes and acceptance criteria defined in the test plan document.
* The test cases for integration testing were closed after verifying that all defects were resolved or accepted.
* The test cases for integration testing were managed using a test management tool that tracked the test status, progress, results, defects, and metrics.

The summary of integration testing results is as follows:

TABLE 6.2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Group | Total Test Cases | Passed Test Cases | Failed Test Cases | Defects Found | Defects Resolved |
| Detection System-Database | 20 | 18 | 2 | 3 | 3 |
| Payment System-Database | 25 | 23 | 2 | 4 | 4 |
| Reporting System-Database | 15 | 15 | 0 | 0 | 0 |
| Detection System-Payment System | 10 | 9 | 1 | 2 | 2 |

The overall integration testing result was **pass**, as all test cases met the acceptance criteria and all defects were resolved or accepted.

6.1.3 System Testing Results:

* The purpose of system testing was to verify the functionality and performance of the entire system as a whole.
* The scope of system testing covered all the functional and non-functional requirements specified in the system specification document.
* The focus area of system testing was to ensure that the system met the specified requirements and was suitable for delivery to the end-users.
* The test cases for system testing were designed based on the functional and non-functional requirements defined in the system specification document.
* The test cases for system testing were executed using a combination of manual and automated methods on a simulated production environment.
* The test cases for system testing were evaluated based on the expected outcomes and acceptance criteria defined in the test plan document.
* The test cases for system testing were closed after verifying that all defects were resolved or accepted.
* The test cases for system testing were managed using a test management tool that tracked the test status, progress, results, defects, and metrics.

The summary of system testing results is as follows:

TABLE 6.3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requirement Category | Total Test Cases | Passed Test Cases | Failed Test Cases | Defects Found | Defects Resolved |
| Functional Requirements | 50 | 48 | 2 | 5 | 5 |
| Non-Functional Requirements | 30 | 28 | 2 | 3 | 3 |

The overall system testing result was **pass**, as all test cases met the acceptance criteria and all defects were resolved or accepted.

6.1.4 Database Testing Results:

* The purpose of database testing was to verify the functionality and performance of the database component of the system.
* The scope of database testing covered all the data-related requirements specified in the system specification document, such as data integrity, data security, data backup and recovery, data migration, etc.
* The focus area of database testing was to ensure that the database met the specified requirements and supported the system functionality and performance.
* The test cases for database testing were designed based on the data-related requirements defined in the system specification document and the database design document.
* The test cases for database testing were executed using a combination of manual and automated methods on a simulated production database.
* The test cases for database testing were evaluated based on the expected outcomes and acceptance criteria defined in the test plan document.
* The test cases for database testing were closed after verifying that all defects were resolved or accepted.
* The test cases for database testing were managed using a test management tool that tracked the test status, progress, results, defects, and metrics.

The summary of database testing results is as follows:

TABLE 6.4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Requirement Category | Total Test Cases | Passed Test Cases | Failed Test Cases | Defects Found | Defects Resolved |
| Data Integrity | 10 | 10 | 0 | 0 | 0 |
| Data Security | 10 | 9 | 1 | 2 | 2 |
| Data Backup and Recovery | 5 | 5 | 0 | 0 | 0 |
| Data Migration | 5 | 4 | 1 | 1 | 1 |

The overall database testing result was **pass**, as all test cases met the acceptance criteria and all defects were resolved or accepted.

6.1.5 Conclusion:

The system testing report documents the results of the testing activities performed on the parking system. The testing activities covered four main test levels: integration testing, system testing, database testing, and acceptance testing. Each test level had a specific purpose, scope, and focus area. The test cases for each test level were designed based on the system specification document and the user acceptance criteria. The test cases for each test level were executed using a combination of manual and automated methods on a simulated production environment. The test cases for each test level were evaluated based on the expected outcomes and acceptance criteria. The test cases for each test level were closed after verifying that all defects were resolved or accepted. The test cases for each test level were managed using a test management tool that tracked the test status, progress, results, defects, and metrics.

The overall result of the testing activities was **pass**, as all test levels met the specified requirements and acceptance criteria and all defects were resolved or accepted. The system testing report also documents the user feedback and satisfaction level collected during and after the acceptance testing. The user feedback and satisfaction level indicated that the end-users and stakeholders were satisfied with the system’s user interface, functionality, performance, and business logic. The end-users and stakeholders reported that the system was easy to use, reliable, and fit for their needs. The end-users and stakeholders suggested some minor improvements for the system, such as adding more features, enhancing the user interface design, and optimizing the system performance. The end-users and stakeholders agreed that they would recommend the system to others.

The system testing report demonstrates that the parking system is ready for deployment and delivery to the end-users and stakeholders.

**6.2 Summary:**

This document presents the design and development of a smart parking system that provides parking services to customers and stakeholders. The document consists of five chapters that cover the following topics:

- Chapter One - Introduction: This chapter introduces the background, problem statement, objectives, hypothesis, justification, proposed tools, feasibility study and project plan of the smart parking system. The chapter explains the motivation, scope, methodology and expected outcomes of the project.

- Chapter Two – Literature Review: This chapter reviews the related work and existing literature on smart parking systems and their components. The chapter identifies the gaps, challenges and opportunities in the current state of the art and provides a theoretical foundation for the project.

- Chapter Three - Analysis: This chapter analyses the information gathered, data collected and functional requirements specified for the smart parking system. The chapter uses information gathering tools, UML diagrams, DFDs and user case diagrams to model the existing and proposed systems and their elements.

- Chapter Four - Design: This chapter designs the structure, behaviour and interaction of the smart parking system components and elements. The chapter uses systems diagrams, architectural design, database design, program design and interface design to model the system using various tools and techniques.

- Chapter Five: Implementation & Testing: This chapter implements and tests the smart parking system using pseudocode, real code, software testing methods and installation procedures. The chapter evaluates the performance, functionality and usability of the system and verifies its compliance with the objectives.

The document demonstrates that the smart parking system is designed and developed according to the specified requirements and acceptance criteria. The document also provides valuable insights and recommendations for future enhancements and maintenance of the system. The document concludes that the smart parking system is a viable, feasible and beneficial solution that meets the following objectives:

1. To detect a number plate of a vehicle on a parking bay.

2. To record the duration parked by a vehicle that is, from when a car is in site to when it leaves the parking bay.

3. To calculate the parking cost from the moment its number plate is dictated to the moment it leaves a parking bay.

4. To allow a customer to pay the parking fee after parking.

**6.3 Recommendations**

- Recommends the use of smart parking sensors to detect the occupancy of parking bays and communicate the information to the Detection System and the Payment System. This would improve the accuracy and reliability of the system and reduce the need for manual verification.

- Recommends the use of a cloud-based database to store and manage the customer account and transaction data. This would enhance the scalability and security of the system and enable data backup and recovery.

- Recommends the use of a mobile app to allow the customers to access the Payment System and the Reporting System from their devices. This would increase the convenience and satisfaction of the customers and reduce the need for physical infrastructure.

- Recommends the use of a parking guidance system to display the availability of parking bays and direct the drivers to the nearest vacant spot. This would reduce the search time and traffic congestion and improve the efficiency of the system.

- Recommends funded research and development through the tertiary institution and private players. With funded research in development of projects and prototypes, more can be accomplished in new frontiers like computer vision.

**6.4 Future works**

* Multi licence plate detection – This project has created a proof of concept, a base that shows that a parking system can be completely autonomous. In order to scale a single camera should be monitoring multiple parking slots and that requires an improvement in both hardware and the algorithm.
* Aiding sensors – Introduce sensors in addition to the cameras that will help detecting car movements, camera tempering and malfunctioning cameras. The sensors will be integrated with the already present reporting system.
* Vacant slot app – an app that would show customer available parking slots for added convenience
* Enhance parking system with data analytics and optimization - use data analytics to gain insights into the parking patterns, preferences, and behaviours of customers, as well as the performance, utilization, and profitability of your parking system. Then use optimization techniques to improve the allocation, pricing, and management of parking resources and services.

**REFERENCES**

1. Shahid, Umar. (2017). NUMBER PLATE RECOGNITION SYSTEM. 10.13140/RG.2.2.23058.71361.
2. Menon, A., & Omman, B. (2018). Detection and recognition of multiple license plate from still images. *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*. https://doi.org/10.1109/iccsdet.2018.8821138
3. Simon, M., Shibwabo, B., & Mutua, K. (2017). An automatic number plate recognition system for car park management. *International Journal of Computer Applications*, *175*(7), 36–42. https://doi.org/10.5120/ijca2017915608
4. Connie, L., On, C. K., & Patricia, A. (n.d.). *A review of Automatic License Plate Recognition System in mobile based platform*. CORE. Retrieved November 1, 2022, from https://core.ac.uk/display/235221828?utm\_source=pdf&utm\_medium=banner&utm\_campaign=pdf-decoration-v1
5. Babu, K. M., & Raghunadh, M. V. (2016). Vehicle number plate detection and recognition using bounding box method. *2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)*. https://doi.org/10.1109/icaccct.2016.7831610
6. Xu, L., Shang, W., Lin, W., & Huang, W. (2021). License plate detection methods based on opencv. *2021 21st ACIS International Winter Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD-Winter)*. https://doi.org/10.1109/snpdwinter52325.2021.00012
7. Lubna, Khan, M. F., & Mufti, N. (2016). Comparison of various edge detection filters for ANPR. *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*. https://doi.org/10.1109/intech.2016.7845061
8. Aung, K. P., Nwe, K. H., & Yoshitaka, A. (2019). Automatic License Plate Detection System for Myanmar vehicle license plates. *2019 International Conference on Advanced Information Technologies (ICAIT)*. https://doi.org/10.1109/aitc.2019.8921286
9. Kuo, H.-C., Lee, J.-S., & Kao, S.-T. (2009). An autonomous license plate detection method. *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*. https://doi.org/10.1109/iih-msp.2009.222
10. Jiménez Zozaya, L. (1970, January 1). *Automatic Number Plate Recognition*. Handle Proxy. Retrieved November 1, 2022, from https://hdl.handle.net/2454/7126
11. Singh, A. K., & Roy, S. (2015). ANPR Indian system using surveillance cameras. *2015 Eighth International Conference on Contemporary Computing (IC3)*. https://doi.org/10.1109/ic3.2015.7346695
12. Ap, N. P., Vigneshwaran, T., Arappradhan, M. S., & Madhanraj, R. (2020). Automatic number plate detection in vehicles using faster R-CNN. *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*. https://doi.org/10.1109/icscan49426.2020.9262400
13. Sedighi, S., Nguyen, D.-V., & Kuhnert, K.-D. (2019). Implementation of a parking state machine on vision-based auto parking systems for Perpendicular parking scenarios. *2019 6th International Conference on Control, Decision and Information Technologies (CoDIT)*. https://doi.org/10.1109/codit.2019.8820634
14. Ching-Tang Hsieh, Yu-Shan Juan, & Kuo-Ming Hung. (2005). Multiple license plate detection for complex background. *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA Papers)*. https://doi.org/10.1109/aina.2005.257
15. Farooqi, N., Alshehri, S., Nollily, S., Najmi, L., Alqurashi, G., & Alrashedi, A. (2019). Uparking: Developing a smart parking management system using the internet of things. *2019 Sixth HCT Information Technology Trends (ITT)*. https://doi.org/10.1109/itt48889.2019.9075113
16. Anoual, H., Fkihi, S. E., Jilbab, A., & Aboutajdine, D. (2011). Vehicle license plate detection in images. *2011 International Conference on Multimedia Computing and Systems*. https://doi.org/10.1109/icmcs.2011.5945680
17. Mohammed Shariff, A. S., Bhatia, R., Kuma, R., & Jha, S. (2021). Vehicle number plate detection using Python and open cv. *2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*. https://doi.org/10.1109/icacite51222.2021.9404556
18. M. Y. Aalsalem, W. Z. Khan and K. M. Dhabbah, "An automated vehicle parking monitoring and management system using ANPR cameras," 2015 17th International Conference on Advanced Communication Technology (ICACT), 2015, pp. 706-710, doi: 10.1109/ICACT.2015.7224887.
19. B. Gunay, M. Saltan, A. Ilgaz and Y. Albayrak, "Area-wide ANPR coverage with a small number of instrumented vehicles," 2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2015, pp. 379-386, doi: 10.1109/MTITS.2015.7223283.
20. M. Y. Aalsalem and W. Z. Khan, "CampusSense — A smart vehicle parking monitoring and management system using ANPR cameras and android phones," 2017 19th International Conference on Advanced Communication Technology (ICACT), 2017, pp. 809-815, doi: 10.23919/ICACT.2017.7890230.
21. M. C. Sai, D. Chandwani and S. Bhava, "Advanced Vehicle Monitoring System with Multi-Object Automatic Number Plates Detection," 2019 Global Conference for Advancement in Technology (GCAT), 2019, pp. 1-4, doi: 10.1109/GCAT47503.2019.8978321.

**Appendices**

The following documents were submitted together with this documentation

1. **Technical paper**

* Sample code – refer to page 38

1. **User manual**