TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Khwopa College Of Engineering
Libali, Bhaktapur
Department of Computer Engineering

A FINAL REPORT ON

SMART PARKING MANAGEMENT SYSTEM

Submitted in partial fulfillment of the requirements for the degree

BACHELOR OF COMPUTER ENGINEERING

Submitted by

Ankit Kayastha	KCE076BCT006
Dibas Regmi	KCE076BCT016
Shopnil Shrestha	KCE076BCT042
Shreyas Dhakal	KCE076BCT043

Under the Supervision of
Er. Bindu Bhandari

Khwopa College Of Engineering
Libali, Bhaktapur
2023-24

Copyright

Acknowledgement

We take this opportunity to express our sincere gratitude to the Department of Computer Engineering, Khwopa College of Engineering, for providing us this opportunity to become familiar with real-world projects and providing us with all the necessary guidance to complete this project.

First and foremost, we are deeply thankful to our Head of the Department, Er. Dinesh Gothe, for his continuous support and encouragement throughout this endeavor. We extend our heartfelt appreciation to our project supervisor, Er. Bindu Bhandari, for her invaluable guidance, insightful feedback, and unwavering encouragement.

We are also thankful to Er. Niranjan Bekoju for his valuable input and constructive criticism, which have greatly enriched the quality of this work. Their expertise and encouragement have been instrumental in shaping the direction of this project.

Finally, we would like to acknowledge everyone who directly or indirectly helped in completing this project.

Ankit Kayastha	KCE076BCT006
Dibas Regmi	KCE076BCT016
Shopnil Shrestha	KCE076BCT042
Shreyas Dhakal	KCE076BCT043

Abstract

Navadurga Smart Parking System (NSPS) had been a technology-driven and artificial intelligence based solution that aimed to optimize the management and utilization of parking spaces in urban areas and commercial establishments. The NSPS utilized various technologies to provide efficient and convenient parking services for both drivers and parking operators. The integration of YOLO for parking occupancy detection and CNN for license plate recognition had offered several key advantages. Firstly, it provided real-time insights into parking availability, allowing drivers to quickly locate vacant spaces and optimize their parking experience. Secondly, the automatic identification of license plates had streamlined entry and exit procedures, enhancing the overall efficiency and convenience of the parking facility. This data had been collected and communicated through a network infrastructure, enabling real-time parking space availability and usage monitoring. Drivers had been able to access a mobile application to check the availability of parking spaces in real-time. It also offered reservation capabilities, allowing drivers to reserve parking spaces in advance and enabling cashless transactions for parking payments. By leveraging advanced technologies and data-driven insights, NSPS had streamlined parking operations and made parking more convenient and efficient for drivers. The system had offered a comprehensive solution to address parking challenges in urban areas. Implementation of NSPS had the potential to enhance urban mobility, reduce environmental impact, and improve overall parking experiences.

Keywords: *Artificial Intelligence, Data-driven, Real-time, Technology-driven, YOLO*

Contents

List of Tables

List of Figures

List of Symbols and Abbreviation

AI	Artificial Intelligence
ANNPR	Automatic Nepali Number Plate Recognition
ANPDR	Automatic Number Plate Detection and Recognition
ANPR	Automatic Number Plate Recognition
AWS	Amazon Web Service
BCE	Binary Cross Entropy
CCTV	Closed-Circuit Television
CIoU	Complete Intersection over Union
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
IoU	Intersection Over Union
ML	Machine Learning
NMS	Non-Max Suppression
OCR	Optical Character Recognition
OpenCV	Open Source Computer Vision Library
ReLU	Rectified Linear Unit
RTSP	Real-time Streaming Protocol
SDLC	Software Development Lifecycle
SPMS	Smart Parking Management System
SQL	Structured Query Language
UI	User Interface
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Background

As urbanization continues to accelerate, the demand for efficient and intelligent parking solutions has become paramount. Traditional parking systems often face challenges such as limited parking space utilization, manual billing processes, inadequate monitoring capabilities, and overall inefficiency. Organizations such as schools, universities, and multinational corporations witness heavy inflow and outflow of vehicles throughout the day. Vehicle drivers find it difficult to get real-time information about an available parking slot and manage the parking of the vehicle as the size of the parking slot is getting reduced and vehicle size is increasing. To address these limitations and enhance the parking experience for both vehicle owners and parking facility managers [6], the idea of creating a smart city is now becoming possible with the emergence of the Internet of Things. Smart cities are a current emerging trend that aims to effectively and smartly automate the monitoring, access and usage of the infrastructure underpinning the major services offered to the citizens. Despite the innovation of intelligent transportation systems, the parking management systems have not seen tremendous advances in the way the availability of parking spots is being advertised to drivers and the tools made available to them to easily park and locate their vehicles within large multi-storey car parking. [7]. One of the key issues that smart cities relate to are car parking facilities and traffic management systems [8]. A cloud-based smart parking application will enable real-time parking availability monitoring and reservation thereby providing better services to the end users as well as reducing the workload of the parking administrator [9]. A Smart Parking Management System is proposed that leverages advanced technologies such as number plate recognition and real-time parking space detection. The advent of smart parking management systems offers promising solutions to address the challenges associated with urban parking. With the integration of IoT, data analytics, and machine learning, these systems aim to revolutionize the way parking spaces are managed and utilized.

1.2 Problem Statement

The current state of parking facilities poses significant challenges that impact both vehicle owners and parking operators.

The following key problems are prevalent:

- **Lack of Real-Time Information:** Traditional parking systems are very dependent on manual labor and possess multiple problems such as long time to find available parking slots, a big amount of cost to pay parking staff, and unavailability of parking data [10]. Without real-time monitoring capabili-

ties and logging of user data, it also creates security concerns and increases the risk of fraudulent activities within parking facilities.

- **Manual Billing and Inconvenience:** Traditional parking systems heavily rely on manual billing processes, including the issuance of paper tickets and manual payment collection. These processes are time-consuming, error-prone, and lack flexibility in terms of payment options. Vehicle owners face inconvenience due to the need for cash transactions or limited payment methods, impacting the overall user experience.

- **Lack of Parking Space Monitoring:** Most of the popular parking systems in the world use coins and tokens. However, despite the widespread use of these systems, they cannot be counted as smart parking, as they do not give precise information regarding booking a specific parking space. Usually, such a system relies on counting how many cars have entered the parking area and calculating the difference between this figure and the maximum number of parking spaces to estimate the number of spaces available. These types of systems usually need a person in charge of the location in case something goes wrong. [11].

- **Hardware Issues :** Most of the recently proposed smart parking systems that have been investigated depend on wireless sensor networks which, in reality, need to be placed in small holes in each parking spot, so applying this system involves changing the infrastructure in addition to many complex circuit requirements. Environmental factors such as snow or dust could fail the whole system [11].

1.3 Objective

The primary objective of our proposed Nawadurga Parking System is:

- To optimize parking space information distribution by implementing real-time vehicle and numberplate detection and monitoring via mobile application.

1.4 Scope of the project

Nawadurga Parking System will consist of a mobile application and a backend-python code server. The Mobile Application will be responsible for handling both the mobile view and the table view. It will display the parking slots contained in different zones. Each zone represents a specific parking area. It will also generate the bills and provide an online payment portal to the user. The app will distribute vehicle activity and status data to the user while providing important push notifications. The server will use a Computer Vision algorithm Called YOLOv8 and CNN for backend processing. However, there are some issues shown by these algorithms when used for our parking area. These issues are:

- The CCTV cameras are fitted in nonoptimal positions which generates a slanted view of the parking and the entrance,

- The cars in very close approximation in a side view are hard to classify,
- YOLO sometimes confuses to differentiate between two cars because of non-optimal footage,
- YOLO could not classify the objects whose edges are not well defined and apart from other objects,
- YOLO as well as CNN could not detect objects in harsh lighting conditions of daytime due to high exposure and high contrast footage from the used CCTV. It also may be inefficient during nighttime lighting conditions

Some of these limitations are hardware-based which are out of the scope of this project, e.g. CCTV camera is fitted sidewise. It is very difficult for YOLO to detect all the cars and the CNN to detect the numberplates with side view angle images. One of the other limitations is the inappropriate parking of cars, which is again out of the scope of this project.
Some other issues are the placement of CCTV cameras in the Parking area. As there were only specific ideal locations for the placement of CCTV cameras for the total view of the parking area. Other issues include having only 8 cameras to cover the entire parking area and a single camera to cover the entrance, which is used for numberplate detection.

Chapter 2

Literature Review

2.1 Historical Development

Table 2.1: Review Matrix with Research Papers and summary of corresponding papers.

S.N	Title	Summary
1	Intelligent parking lot application using wireless sensor networks [12]	In this paper, the authors used magnetic sensors and ultrasonic sensors for better accuracy and quality recognition of the cars in the parking space. The authors suggested that the use of wireless sensor networks in parking lot management can lead to significant improvements in efficiency and convenience for drivers.
2	Robust Parking Space Detection Considering Inter-Space Correlation [13]	The paper proposes a method for recognition of the available parking lot by using the Markov Random Field framework and Support Vector Machines. The model is trained to detect the car in the parking space by using a machine learning algorithm rather than segmentation of the vehicle in an image that is taken from the parking lot.
3	Vehicle and Parking Space Detection Based on Improved YOLO Network Model [14]	This paper proposes a vehicle and parking space detection method based on the improved YOLOv3 algorithm with the use of the PKLot dataset. The authors concluded that under the influence of factors such as illumination and weather, it has a certain influence on the detection effect of the algorithm.
4	Vehicle Number Plate Recognition and Parking System [15]	The paper combines the capabilities of hardware and software into an integrated automatic system. Here image processing software locates the plate of the car at the entrance of the parking lot using the camera. The output of image processing is the plate number which decides the entry of the car into the parking lot.

5	Vehicle Detection Using YOLOV3 for Counting the Vehicles and Traffic Analysis [16]	This study analyzed two techniques for vehicle detection namely YOLO V3 and CNN. The study concluded that the efficiency of YOLOv3 is 95 and that of CNN is 85. YOLOv3 tends to be additionally reached out for auto vehicle discovery and traffic surveillance.
6	Car parking occupancy detection using smart camera networks and Deep Learning [17]	This study presented an approach for real-time car parking occupancy detection that uses a Convolutional Neural Network (CNN) classifier running on-board a smart camera with limited resources. It proposes and evaluates an approach for using Convolutional Neural Networks to classify the parking space occupancy directly onboard a Raspberry Pi camera.
7	Implementation of Smart Parking System Using Image Processing [18]	This study utilizes the YOLOv3 algorithm to detect the coordinates of both parking lots and parked vehicles separately. To train and evaluate the model, the PKLot database was used as the dataset and tested the model's performance under different weather conditions. The proposed model achieved an average performance of 88.0%, with the highest performance demonstrated on sunny days and the lowest performance recorded on rainy days.
8	Automatic Parking Management System and Parking Fee Collection Based on Number Plate Recognition [19]	This paper discussed automatic parking system and electronic parking fee collections based on vehicle number plate recognition without the hassles of using magnetic cards.
9	A Review of Parking Slot Types and their Detection Techniques for Smart Cities [20]	The paper proposed the review of the existing parking slot types and their detection techniques. The study concluded that the majority of smart parking systems in smart cities rely on sensors which drive up the cost of the system. Integrating automation and smart technologies into parking management has huge potential for improving urban mobility and easing congestion.

10	IntelliPark -Smart Parking System [21]	The study proposes a Smart Parking System, which employs a model that works similar to YoloV5 and observes the incoming cars and the parking area, redirecting the incoming cars to the empty spots which would be the desired result.

2.2 Related Work

Table 2.2: Summary of Related Works

Project Title	**Programming languages Used**	**Use of AI**	**Remarks**
Intelligent Parking System	HTML, asp.NET	No	Extracts information from VMS system and simple webpage to display parking availability.
University Car Parking Application	Java, Xaml, Json	No	Android app with QR code scanners and generators
Application-based Smart Parking System	Arduino C++, Java, Xaml, NodeJs	No	Sensor-based approach consisting of microcontrollers, ultrasonic sensors,etc
Smart Parking System for the University of Kufa	Arduino C++	No	Use of ultrasonic sensors and Arduino Uno
Smart Parking System For UF Campus	Html, CSS, PHP, Java, Xaml	Yes	Smart Parking Application using real-time parking space availability and prediction.

2.3 YOLO

YOLO stands for "You Only Look Once," and it is a real-time object detection algorithm developed by Joseph Redmon. [22] YOLO is known for its fast speed and high accuracy, and is a popular choice for object detection in real-time applications such as self-driving cars, surveillance systems.

2.3.1 Residual blocks

First, the image is segmented into several square grids. Each grid cell has a dimension of S x S. Every grid cell will predict B boundary boxes and determine the confidence score for each box to detect objects that appear within them.

2.3.2 Bounding box regression

The next step is to determine the bounding boxes which correspond to rectangles highlighting all the objects in the image. Every bounding box in the image consists of the following attributes: probability score of the grid containing an object(pc), Width (bw), Height (bh), Class (c), and Bounding box center (bx, by).

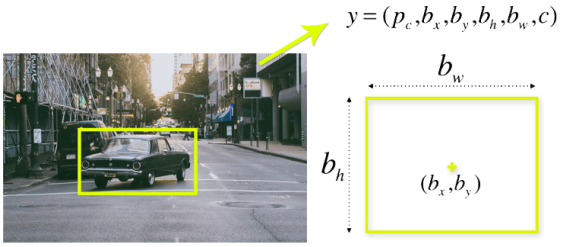


Figure 2.1: Bounding box regression [1]

2.3.3 Intersection Over Union

Intersection over union(IOU) is a measure of overlap between two bounding boxes. In YOLOv8, IoU is used to measure the accuracy of the predicted bounding boxes for objects in an image.
The IoU is calculated by dividing the area of intersection between the predicted bounding box and the ground truth bounding box by the area of union between the two bounding boxes.

In YOLO, a threshold value of IoU is set to determine whether the predicted bounding box is considered as a correct detection or not. If the IoU value is greater than the threshold, the detection is considered correct.
Informally, IoU measures how equal two areas are. In terms of size and location of the area. If two areas are exactly equal, IOU will be 1. If two areas are far apart, even if their shape is the same, they will have IOU 0. And if two areas lie at the same location but their size differs a lot, then also IOU will be a small value. [**?**]

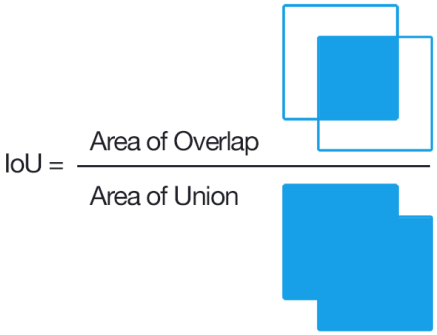


Figure 2.2: Intersection Over Union [2]

Figure 2.3: Goodness measure of IOU [3]

$$IOU(Box1, Box2) = \frac{Intersection_Size(Box1, Box2)}{Union_Size(Box1, Box2)} \tag{2.1}$$

2.3.4 Non-Maximum Suppression

Non-maximum suppression (NMS) is a post-processing algorithm used in object detection tasks to eliminate redundant detections of the same object. It works by selecting the highest-scoring detection (i.e., the detection with the highest confidence score) and suppressing all other detections that have a high overlap (i.e., high IoU) with the selected detection.
The purpose of NMS is to eliminate redundant detections and to select the most accurate detection for each object. This improves the accuracy of the object detection algorithm and reduces false positives.
Input: A list of Proposal boxes B, corresponding confidence scores S, and overlap threshold N.
Output: A list of filtered proposals D.
Algorithm:

A. Sort the detected bounding boxes based on their confidence scores (i.e., probability of containing an object).

B. Select the bounding box with the highest confidence score as the first detection.

C. Calculate the IoU of the selected detection with all other detections.

D. Remove all detections that have a high overlap (i.e., IoU greater than a predefined threshold) with the selected detection.

E. Repeat steps B to D until no more detections are left.

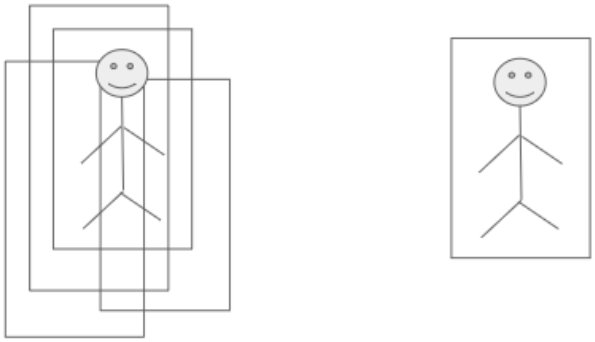


Figure 2.4: Non Maximum Suppression

2.3.5 YOLOv8

In the project, all the YOLO models used are YOLOv8. The reasons for choosing the model among all the others are:

- Delivers superior performance capabilities in comparison to its predecessors.
- Its developer-convenience features, make it easy to use and customize.
- Comes with a command-line interface (CLI) that simplifies the workflow.
- Contains a large and growing community, which provides easy support during technical and operational difficulties.

2.3.5.1 Components

- **Backbone**: The backbone, also known as the feature extractor, is responsible for extracting meaningful features from the input.
- **Neck**: The neck acts as a bridge between the backbone and the head, performing feature fusion operations and integrating contextual information. The Neck assembles feature pyramids by aggregating feature maps obtained by the Backbone, in other words, the neck collects feature maps from different stages of the backbone.
- **Head**: The head is the final part of the network and is responsible for generating the network's outputs, such as bounding boxes and confidence scores for object detection.

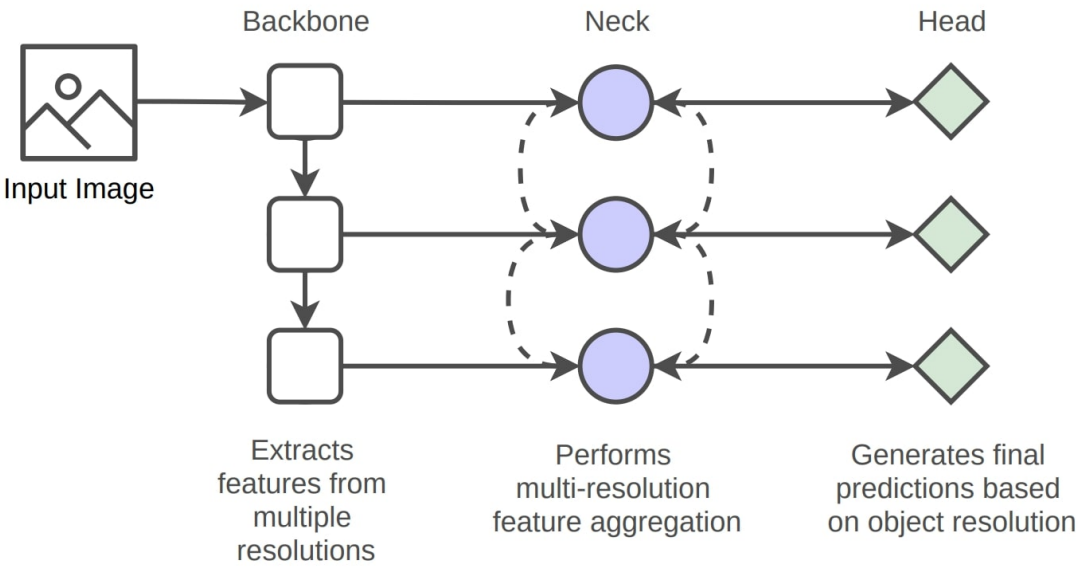


Figure 2.5: Basic Architecture of YOLO [4]

Chapter 3

Feasibility Study

3.1 Economic Feasibility

The total expenditure of the project was just computational power and a few hardware devices like CCTV cameras and an NVR. The dataset required for the project was easily prepared on our own. The computational resource cost included processing power and electricity, which was negligible. The hardware was already available to us. So, the project was economically feasible.

3.2 Technical Feasibility

The necessary technology, such as cameras, communication networks, and software platforms for implementing the system was easily available as one of the members of the project group owned a private parking space. We carefully assessed the system's compatibility with existing infrastructure and parking facilities.

3.3 Operational Feasibility

Upon assessing the impact of the smart parking management system on existing operations and processes, we evaluated whether the proposed system was compatible with the parking facility's operational requirements, rules, and regulations. And there was no possibility of any potential disruptions. The system could be operational just after training the network model. Thus, the project was operationally feasible.

3.4 Time Feasibility

The project was expected to be completed and was completed within the time frame described in the Gantt Chart, considering unexpected anomalies that could arise. The major time was spent on developing the model and training it.

Chapter 4

Requirement Analysis

4.1 Functional Requirement

Functional requirements inform about the basic functionality of the product. Function requirements tell the usage of the system and how the system works. The functional requirements of NSPS are as follows:

4.1.1 Vehicle and Parking Space Status

The system should provide real-time information and logs of the status of the vehicle and parking space. It should provide the all necessary information to the user for making the entire operation of the parking management fully automated.

4.1.2 Unauthorized Entry Detection

The system should be able to detect any form of unauthorized entry in the private parking space and alert the operators through the number plate recognition system.

4.1.3 Payment Integration

Users should be able to make payments for parking through the system. Integration with payment gateways or mobile payment platforms allows for convenient and cashless transactions, supporting various payment methods.

4.2 Non-functional Requirement

These requirements encompass various attributes and qualities beyond the specific functionalities.

4.2.1 Full automation

The system should not require any human interaction being an automated independent running system.

4.2.2 Cost-effective

The system should bear a minimal cost to implement with the required cost coming down just fulfill CCTV installation and cloud computing.

4.2.3 Performance

The system should exhibit fast response times, minimal latency, and high availability to handle a large number of users and parking requests simultaneously. It should be able to process real-time data, update parking space availability promptly, and deliver quick navigation and reservation functionalities.

4.2.4 Reliability

The system should be highly reliable, minimizing downtime and ensuring uninterrupted service. It should have backup mechanisms, redundancy measures, and disaster recovery plans to handle any system failures or disruptions.

4.2.5 Usability

The system should be designed with a user-centric approach, ensuring ease of use and intuitive interactions. It should have a user-friendly interface, clear navigation, and straightforward functionalities to enhance user experience and minimize the learning curve.

Chapter 5
System Design and Architecture

5.1 Use Case Diagram

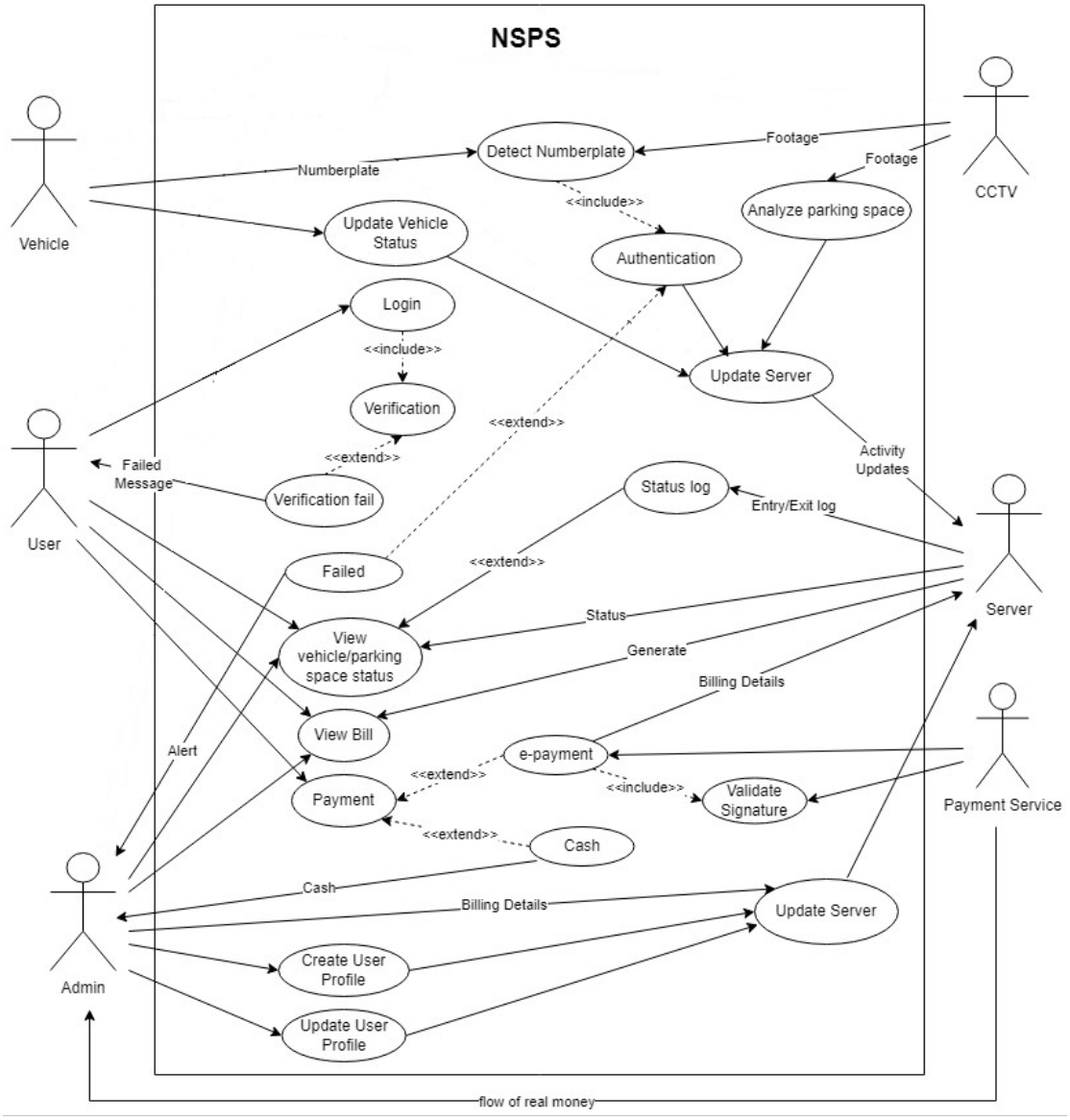


Figure 5.1: System Use Case Diagram

Actors:

- User: Represents individuals using the mobile app to check vehicle and parking status, view logs, and pay bills.
- Admin: Represents the administrator responsible for adding new users and managing user details.
- Server: Represents the central server managing parking space and vehicle information, receiving data from CCTV footage, generating bills, generating logs, and controlling the gate.
- CCTV: Represents the cameras used for capturing footage.
- Vehicle: Represents the vehicles that provide necessary data like position and number plates.
- Payment Service: Represents the online payment portals.

Use Cases:

- Analyze and Update Parking Space and Vehicle Status: The server analyzes and updates parking space and vehicle status based on CCTV footage.
- Detect Numberplate and Inform Server: The CCTV system detects license plates and informs the server. If authentication fails, the server takes appropriate action.
- View Vehicle and Parking Status with Entry/Exit Logs: Users can view their vehicle and parking space status along with entry and exit logs using the mobile app.
- Pay Bills: Users can pay bills generated by the system through the mobile app.
- Add New User and User Details: Admin can add new users and manage user details in the system.

5.2 Design Flow

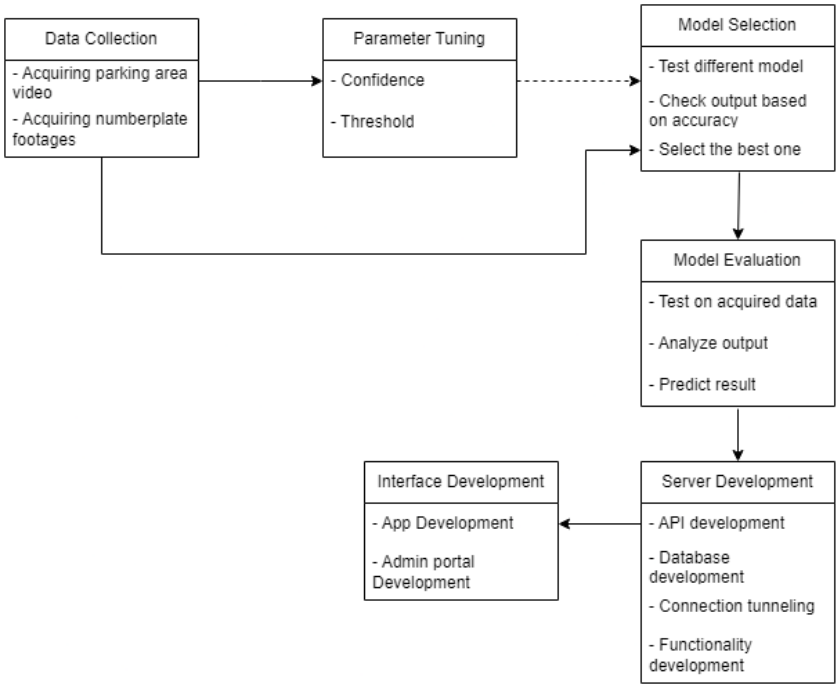


Figure 5.2: Design Flow

The design flow of the parking system is given above. It has all the modules that are necessary to build a smart parking system.

- In the first module, it has a data-acquiring step in which the data is gathered. Data gathering is the basic starting point for any product to be accurate.
- The subsequent step was the model selection. In the model selection phase, we first studied the best computer vision algorithms around the globe which were tested and compared based on speed and accuracy.
- Finally, through hit and trial and several model testing iterations, we selected YOLOv8n due to its performance and lightweight characteristics for vehicle detection and we constructed ANPDR for numberplate recognition.
- The parameters of the algorithms were tuned based on the accuracy of the models. After each parameter tuning, both models were evaluated. This process was iterated several times. The model was trained on the datasets and, if needed, the parameters and datasets were tuned to get an accurate model. After several iterations, both of the models were able to perform accurately for our project.
- The next step was developing the app. API and database were created for live connections between the app and the backend code. The database and API were created to send the output from the models' code to the server.
- After that, the next phase started designing the interface. The interface is compatible with Android users and iOS users.

5.3 System Workflow

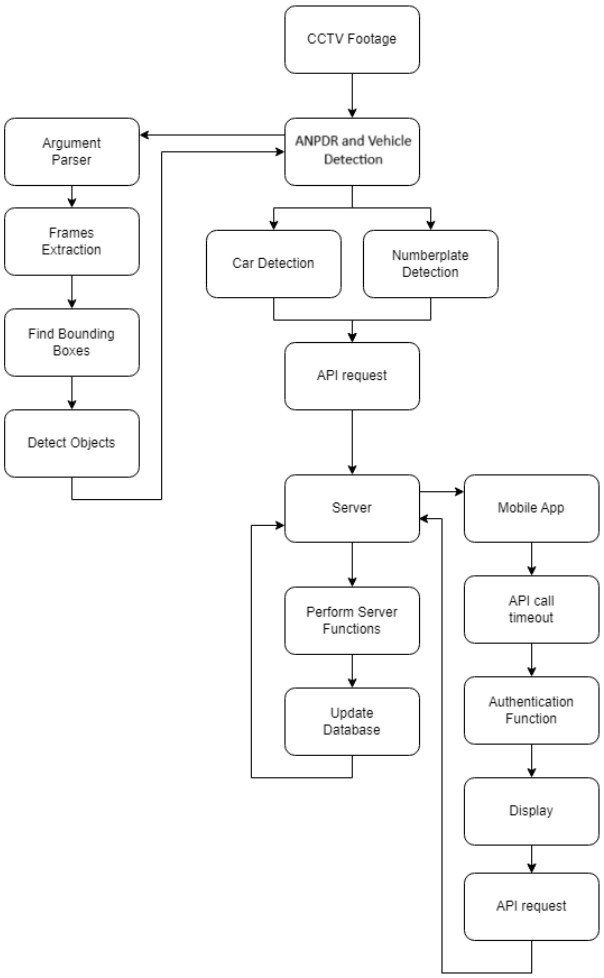


Figure 5.3: System Workflow

The system workflow consists of the following steps:

- The CCTV footage is first captured. Then the footage is loaded into the ANPDR and vehicle detection program.
- The program parses the argument, extracts the frames, finds the bounding boxes, and detects the objects.
- The program also sends the API requests to the server. The server performs its functions, updates the database with new data, and also updates the mobile application.
- The mobile application calls the APIs, performs authentication functions, displays the parking availability and vehicle numberplates as well as sends API requests back to the server.

5.4 Architecture of ANDPR

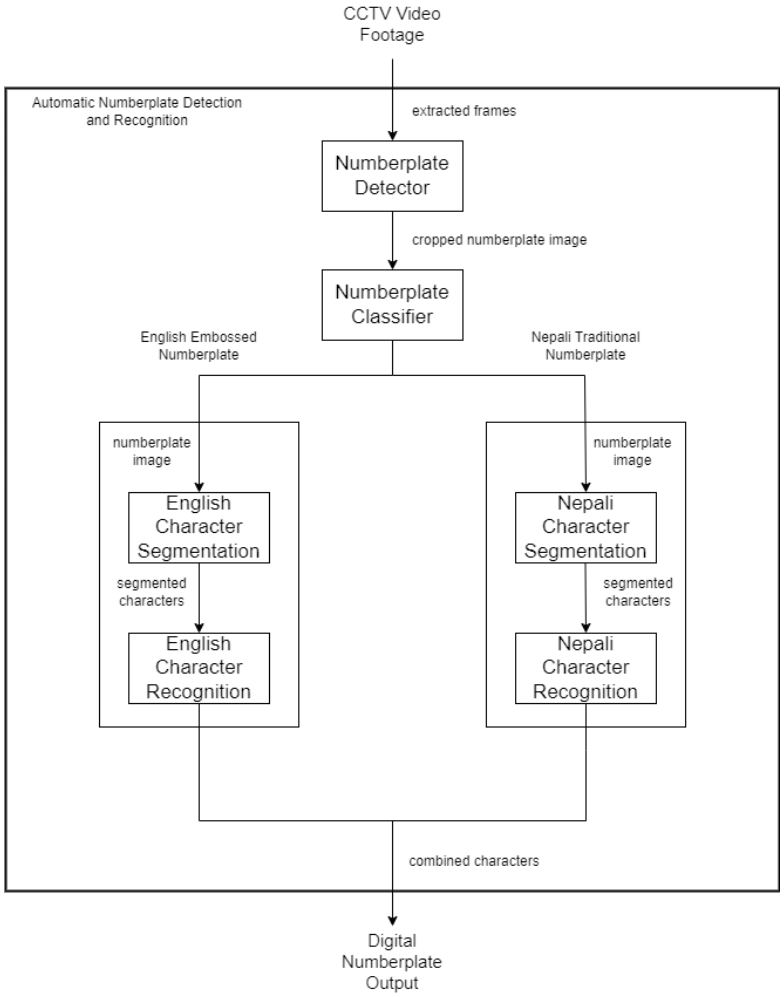


Figure 5.4: Architecture of ANDPR model

The architecture of ANDPR model is described below:

- The CCTV video footage is captured and sent to the numberplate detector model. The model detects the numberplate and then the numberplates are sent further to the numberplate classifier model to classify the type.
- The numberplate classifier classifies the identified numberplates and classifies them into Nepali or English embossed numberplates.
- According to the type of numberplate, the frames are loaded into either Nepali or English character segmentation model which segments the characters.
- Now, these segmented characters are then recognized by using either English Character Recognition model or the Nepali Character Recognition model. The characters are then combined to get the final digital number plate characters.

5.5 Sequence Diagram for Automatic Number-plate Detection and Recognition System

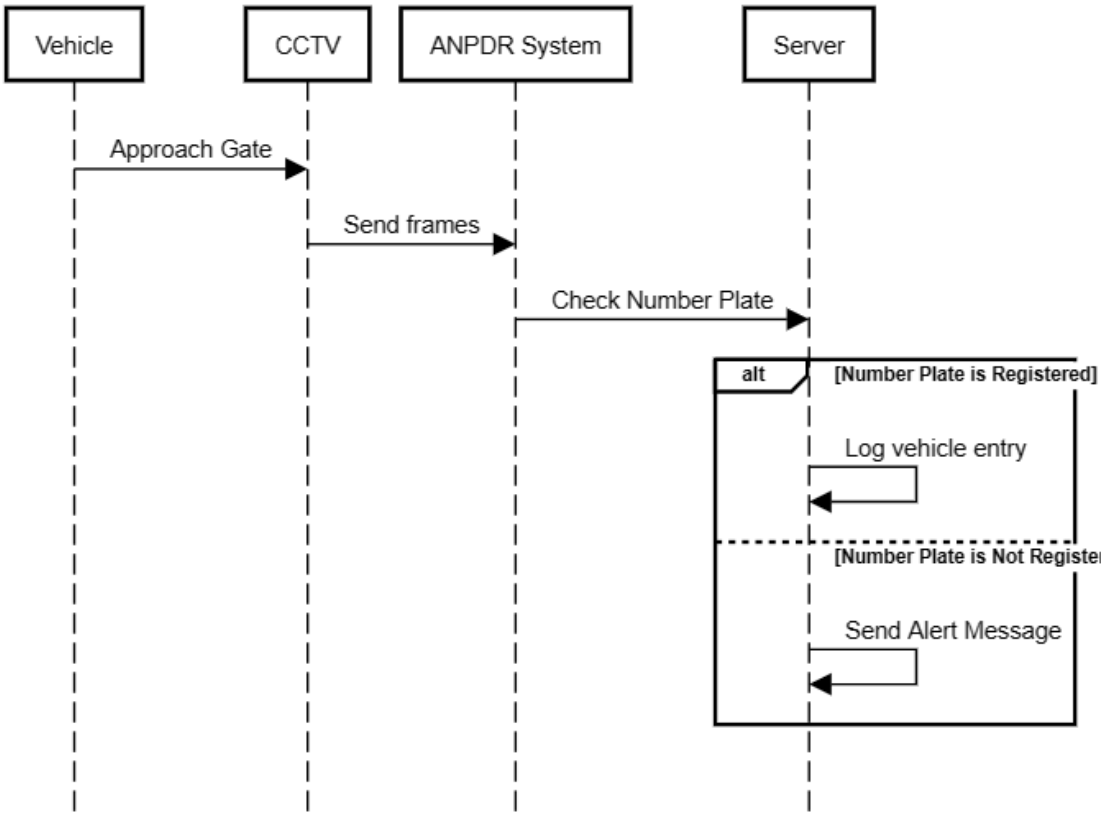


Figure 5.5: Sequence Diagram for ANPDR System

This sequence illustrates the interaction between the vehicle, CCTV, ANPDR system, and server in an ANPDR system deployment. The components of the system are the vehicle, CCTV, ANPDR system, and the server. The steps of the sequence diagram are shown below:

- In the ANPDR sequence diagram, the process begins with a vehicle approaching the gate, which is detected by a CCTV.
- The CCTV then sends captured frames to the ANPDR system for number plate recognition.
- Subsequently, the ANPDR system checks the number plate against the database on the server. Depending on whether the number plate is registered or not, two alternative scenarios unfold.
- If the number plate is registered, the server logs the vehicle entry. However, if the number plate is not registered, the server sends an alert message.

5.6 Sequence Diagram for Vehicle Occupancy Detection System

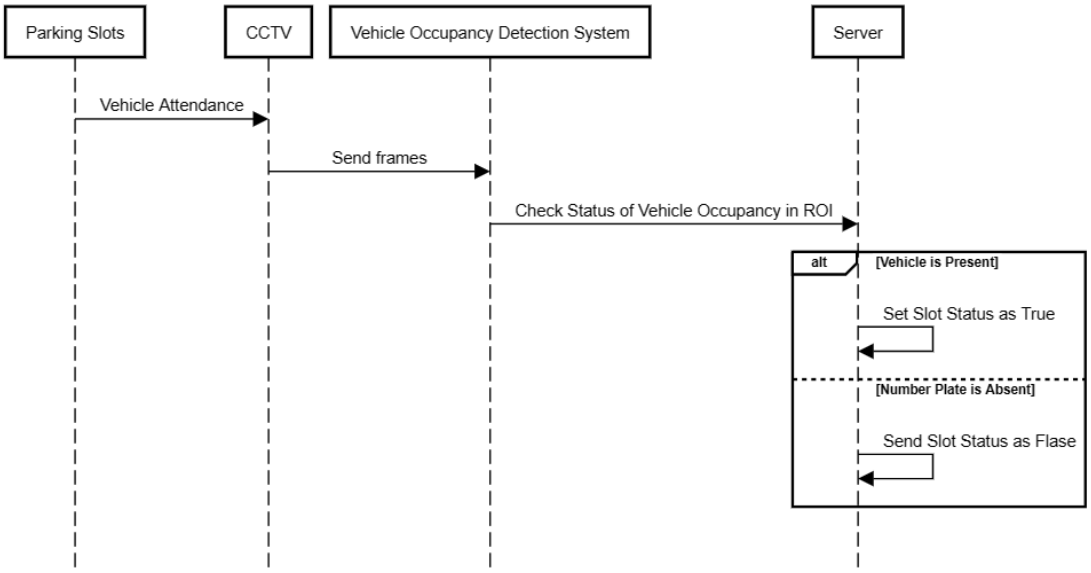


Figure 5.6: Sequence Diagram for Vehicle Occupancy Detection System

This sequence illustrates the interaction between parking slots, CCTV, Vehicle Occupancy Detection System, and the server in the system development. It contains the following components:

- Parking Slots
- CCTV
- Vehicle Occupancy Detection System
- Server

The steps of the sequence diagram are given below:

- In the Vehicle Occupancy Detection System sequence diagram, the process begins with the parking slots that may contain or not contain vehicles parked.
- The CCTV sends the frames of each of these parking slots to the detection system.
- The detection system checks the vehicle occupancy and detects if there are vehicles present in the parking slots or not. Depending on the presence or absence of vehicles, two scenarios occur.
- If the vehicle is present in the slot, the Slot Status in the database is set to True. If there is no vehicle detected in the parking slot, the Slot Status is set to False in the database.

5.7 Entity - Relationship Diagram

The different relationships between the various entities in our system are represented in the diagram.

- Users: This table stores information about users of the parking system. It includes fields for user ID, name, address, phone number, password, etc.
- Vehicle: This table records details about the vehicles registered in the system. It includes fields for vehicle ID, model, number plate, and foreign key.
- Notifications: This table is for managing notifications sent to users. It contains the notification ID, title, subtitle, date of the notification, and the user id to whom the notification is sent.
- Notification tokens: Linked to the Notifications table, this table holds the push notification tokens for users. It has fields for token ID, push notification token and user id.
- Parking slots: This table keeps track of the status of parking slots. It includes fields for slot ID, the status of the slot (slotStatus as a Boolean value), slot number, and a foreign key.
- Vehicle entry exit stamps: This table logs entry and exit times of vehicles to and from the parking. It includes entry time, exit time, and an additional field countFalseSlotStatus possibly to track incorrect slot status reports.
- Parking logs: It logs detailed information about parking events, including the ID, vehicle id, parking space, entry time, and exit time.

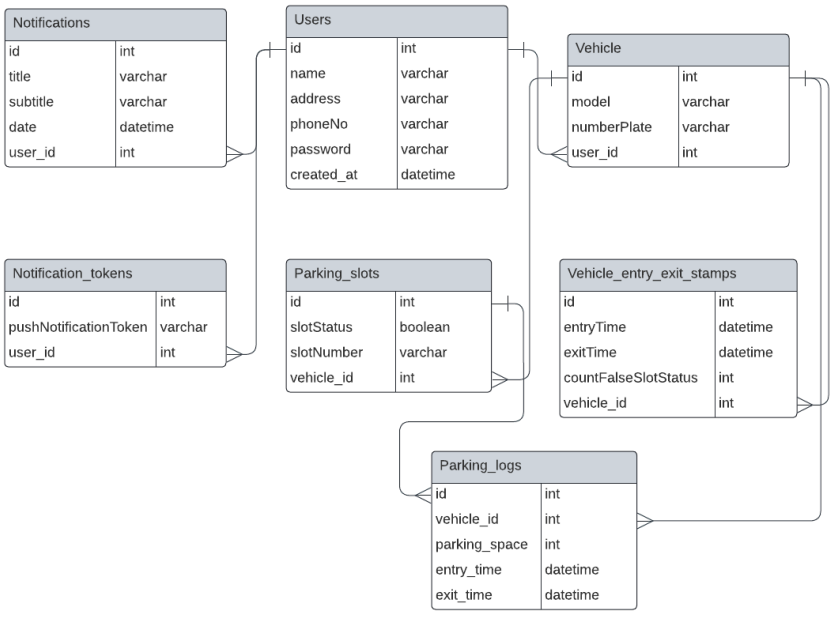


Figure 5.7: Entity Relationship Diagram

Chapter 6

Methodology

6.1 Software Development Approach

The AGILE model is an iterative and flexible software development methodology emphasizing collaboration, adaptability, and customer feedback. It involves breaking down the development process into small increments called iterations or sprints, with each iteration delivering a functional piece of the software.

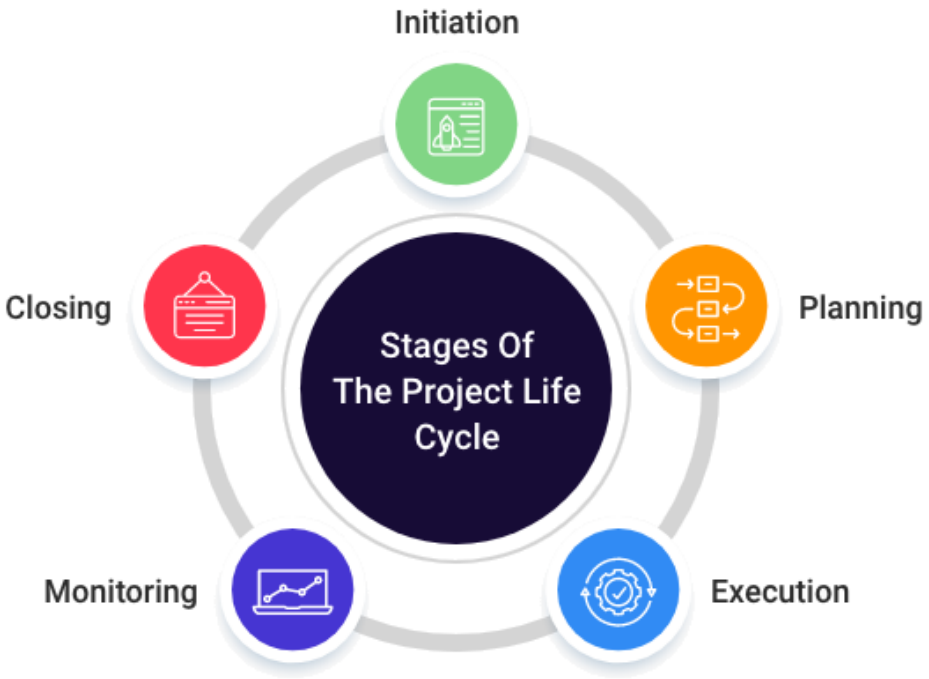


Figure 6.1: SDLC Workflow [5]

The AGILE model consists of the following sequential phases:

- Initiation
- Planning
- Execution
- Monitoring
- Closing

6.1.1 ClickUp as Project Management Tool

ClickUp was utilized as a project management tool for organizing, distributing, managing, and tracking all work, as illustrated in Figure **??**. Tasks were generated as depicted in Figure **??**. Initially, a project plan was formulated and subsequently divided into sprints, during which tasks from the project plan were executed.

6.1.2 Product Backlog

The product backlog is a prioritized list of work items that need to be accomplished to deliver a product in the Scrum framework. It served as an integral planning tool within Scrum methodology, aiding the iterative and incremental software development activities. The project supervisor used to attend the sprint planning meeting with the prioritized agile product backlog and describe the top items to us. Then it was used to determine which items could be completed during the coming sprint. The items were then moved from the product backlog to the sprint backlog. In doing so, each Scrum product backlog item was expanded into one or more sprint backlog tasks so they could more effectively share work during the sprint. The product backlog included the following features:

1. Research and Analysis
2. Data Sourcing and Acquisition
3. Study and Research on Vehicle Detection and OCR Models
4. Selection of the Models
5. Data Cleaning, Preprocessing and Augmentation
6. Experiment with different model configurations and hyperparameters
7. Model Selection and Development
8. Evaluation of the model performance
9. Model Optimization and Enhancement
10. UI design
11. Frontend Development
12. Backend Development
13. Integration and Testing
14. System Evaluation and Visualization
15. Deployment in mobile app

Following tasks were performed during project development:

6.1.2.1 Sprint 1-2: Research and Analysis

During these initial sprints, the focus was on laying the groundwork for the project. Extensive research was implied to understand the details of smart parking systems and existing detection techniques. Simultaneously, various existing detection models were tested to identify the most suitable one for our project's objectives. These sprints were crucial for establishing a solid foundation upon which we built our smart parking management system.

1. Conducted thorough research on object detection and character recognition techniques.
2. Studied and analyzed various existing models suitable for vehicle detection and OCR.
3. Selected the most appropriate model for detection based on research findings.

6.1.2.2 Sprint 3-4: Data Sourcing and Preprocessing

In these sprints, the primary goal was to gather necessary data for training and testing our models. We collected datasets from CCTV footage and reputable sources like Kaggle and ensured they were clean and properly formatted. Data was properly analyzed and minor preprocessing was performed to prepare the datasets for model training.

1. Sourced and acquired datasets from reliable sources like Kaggle, and Roboflow, as well as CCTV footage
2. Analyze and preprocess the collected datasets

6.1.2.3 Sprint 5-6: Model Development and Evaluation

In these sprints, our focus then shifted to model development consisting of the following tasks:

1. Explore various deep learning architectures suitable for vehicle detection and OCR models (e.g., CNNs, GANs).
2. Experiment with different model configurations and hyperparameters.
3. Train initial models using labeled data.
4. Evaluate model performance using appropriate metrics (e.g., accuracy, precision, recall).

6.1.2.4 Sprint 7-8: Model Optimization and Enhancement

With the progress of the project, the objective was shifted to refine and enhance the vehicle detection and numberplate classification models to ensure high accuracy in detecting the objects. Concurrently, the development of the user interface design for the smart parking management system was carried out, ensuring it is intuitive and user-friendly. It consisted of the following tasks:

1. Fine-tune model parameters to improve detection accuracy and reduce false positives.
2. Avoid overfitting of the model by tuning of hyper-parameters.

6.1.2.5 Sprint 9-10: Frontend and Backend Development

Upon optimization of the model, the focus was on the creation of an engaging and responsive frontend interface for users to interact with, while also developing the necessary backend infrastructure to support the system's functionality. It consisted of the following tasks:

1. Implementing the front end of the client-side application for the smart parking management system.
2. Developing the backend, including server-side logic and database integration.

6.1.2.6 Sprint 11-12: Integration, Testing, and Evaluation

In the final stages of development, our priority was to integrate all components of the smart parking management system seamlessly. We performed below mentioned activities:

1. Integrating the training models into a unified system.
2. Developing a user-friendly interface for interacting with the system.
3. Conducting thorough testing to validate system functionality and reliability.

6.1.2.7 Sprint 13-14: Deployment and Finalization

As we approached the end of the project, the focus was on deploying the smart parking management system in a production environment, making it accessible to users. We finalized all project documentation, ensuring it was comprehensive and well-documented. It consisted of:

1. Deploying the smart parking management system in a web application environment.
2. Finalizing project documentation, including system architecture, user manual, and technical specifications.
3. Addressing any remaining issues or feedback.

6.2 Software Used

For the project, the following software were used:

- Amazon Web Service
- ClickUp
- Java
- MySQL
- PyTorch
- Python
- React Native
- Tensorflow
- VS Code

6.3 Hardware Used

For this system, the following hardware were used for development purposes:

- CPU: Ryzen 7 5800x
- GPU: Nvidia RTX 3080ti
- RAM : 16x2 GB 3600mhz
- NVR (Network Video Recorder)
- CCTV Cameras
- Linux Server

6.4 Description of Workflow

6.4.1 Hardware Installation

The primary hardware components required for the system were installed. This process involved mapping the layout of the entire parking area and optimizing the best available and accessible locations for installing the CCTV cameras, NVR and Servers.

After the initial planning and a installation layout, the cameras were installed accordingly and connected to an NVR with the use of power-ready coaxial cables. The NVR was set up to keep the recording stored on a local disk and also network configured to provide a real-time streaming feed using RSTP protocol. The NVR also served as a purposeful Analog to Digital signal converter. Similarly, two CCTV were installed on the inside and outside entrance to monitor and capture the entry of vehicles and their numberplate. The camera was installed in such a position and angle that it captured the best view of the number plate.

Table 6.1: Hardware

Hardware	**Count**
Analog CCTV	7
Varifocal Digital CCTV	2
NVR	1
Linux Server	1
Coaxial Cable	7
Cat6 Cable	2
Switch	1
Router	1

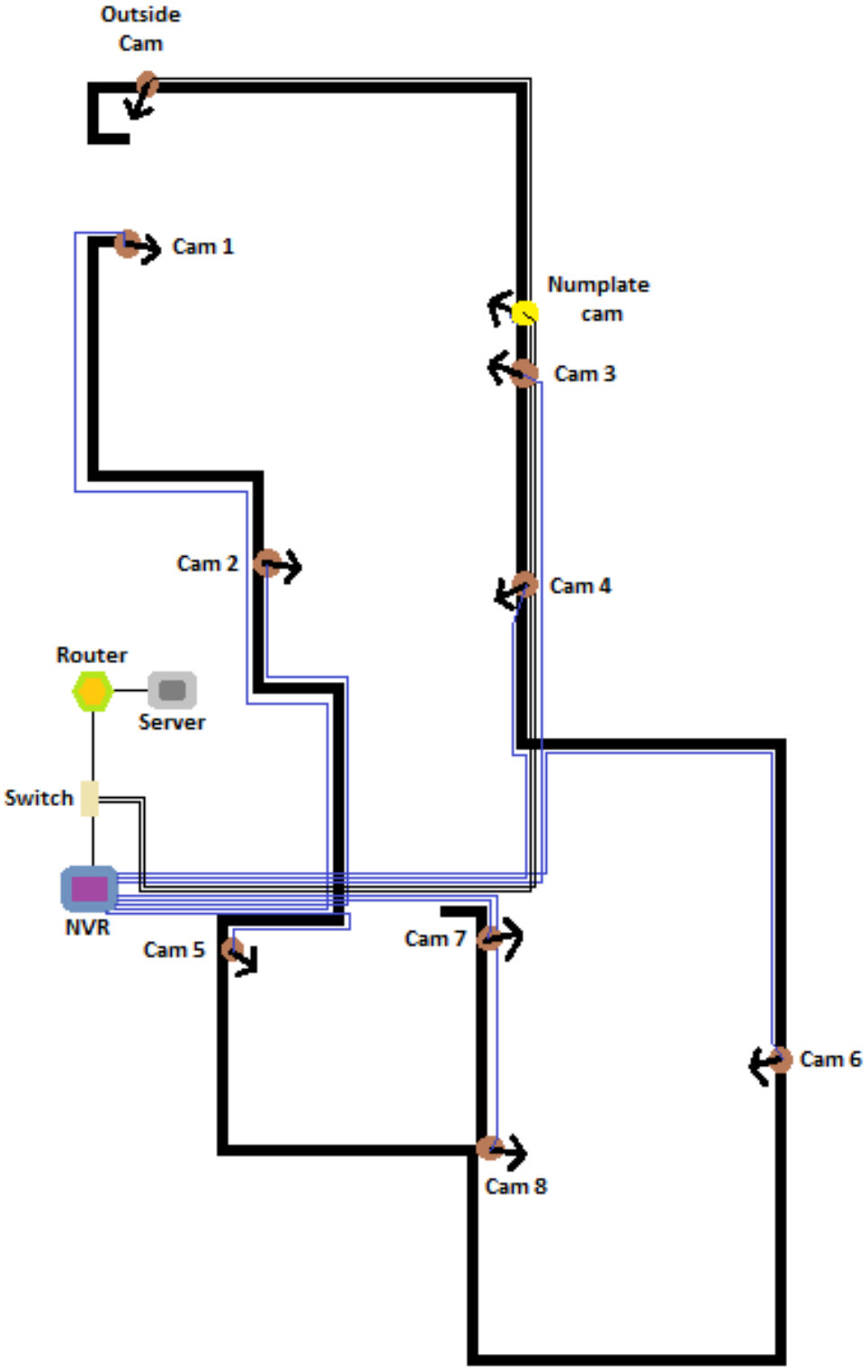


Figure 6.2: Installation Layout

Tasks such as cable routing and cable management were completed as well. A 6 TB hard disk was installed for storing the footage for security purposes. The various devices are IP mapped from a switch to the main home router using Dynamic IP mapping protocol. The recording and streaming configurations were done according to the need and ease of use of the operators. The limited cameras

were rotated and pointed to fixed positions for full coverage of the parking space as needed. This ensured availability of all the image data needed for our models to work.

Figure 6.3: Installation of hardware

6.4.2 Data Collection

The ANPDR architecture comprised of 6 different models and the vehicle detection required a single YOLO model. So as a result, the data collection was a major portion of the project. The different models required different types of datasets. The different sources of dataset for the project were:

- CCTV Footage
- Manual Dataset Collection
- Different Online Sources: Kaggle [23] and Roboflow [24] [25]

The following tables show the total number of data collected for each model.

6.4.2.1 Vehicle Detection Model

For the vehicle detection model, the dataset collected consists of the following types of images as shown in the table:

Table 6.2: Dataset quantities for Vehicle Detection models

Character	**Number of Training Images**	**Number of Validation Images**
Vehicle not covered	2736	684
Vehicle covered	988	247

There were two types of vehicle images collected for the vehicle detection system. In a private parking space, there were numerous vehicles covered with car covers. So, without collection of these images, the detection model was unable to detect these vehicles, resulting in false information to the users. So, collection of images of covered cars were required. Similarly, the lighting conditions were different throughout the day and night. Sometimes, the cars appeared too bright and sometimes the cars could be barely seen. As a result, images under all these conditions were required to be trained for the vehicle detection model.

6.4.2.2 Number Plate Detection and Classification Models

For the numberplate detection and classification models, the dataset collected consisted of the following types of images as shown in the table:

Table 6.3: Dataset quantities for Numberplate Models

Character	**Number of Training Images**	**Number of Validation Images**
Nepali Numberplate	2319	579
Embossed Numberplate	4256	1064

The models required both Nepali as well as Embossed numberplates datasets. The Nepali Numberplate dataset was collected using CCTV footage as well as manual collection from public spaces. For the Embossed Numberplate, there was a limited number of datasets available by manual collection. So we had to collect available online datasets for the model.

6.4.2.3 Nepali Character Segmentation and Classification Models

For the Nepali character segmentation and classification models, the dataset collected consisted of the following types of images as shown in the table:

Table 6.4: Dataset quantities for Nepali models

Character	**Number of Training Images**	**Number of Validation Images**
0	439	109
1	428	107
2	249	62
3	238	59
4	329	82
5	307	76
6	315	78
7	334	83
8	253	63
9	219	54

ba	631	157
cha	455	113
pa	236	59

For the collection of these datasets, we had run the numberplate detection models which were trained from the previous datasets on the collected datasets, which helped us obtain cropped images of the numberplates. From this numberplate, we labeled each character using the above characters. Only Nepali characters like 'ba', 'pa', 'cha' were collected as they were enough for the parking space. For segmentation, we used the same dataset for the recognition but the label of each character was changed to a common label.

6.4.2.4 English Character Segmentation and Classification Models

For the English character segmentation and classification models, the dataset collected consists of the following types of images as shown in the table:

Table 6.5: Dataset quantities for English models

Character	**Number of Training Images**	**Number of Validation Images**
0	1167	291
1	789	197
2	827	206
3	916	229
4	1127	281
5	1180	295
6	1196	299
7	820	205
8	916	229
9	1108	277
A	1167	291
B	1201	300
C	680	170
D	1254	313
E	1124	281
F	664	166
G	760	190
H	1244	311
I	818	204
J	912	228
K	760	190
L	862	215
M	1063	265
N	716	179

O	1072	268
P	1053	263
Q	1204	301
R	677	169
S	811	202
T	982	245
U	650	162
V	1141	285
W	1222	305
X	904	226
Y	808	202
Z	778	194

For the collection of these datasets, we ran the numberplate detection models which were trained from the previous datasets on the collected datasets, which helped us obtain cropped images of the numberplates. From this numberplate, we labeled each character using the above characters. All English characters were collected for this model because of the adequacy of data and requirement. For segmentation, we used the same dataset for the recognition but the label of each character was changed to a common label.

6.4.3 Frame Sampling

After the video footages were captured, the video frames were extracted from the videos using a Python computer vision tool named OpenCV. Python 3.9.13 is used as it was the highest-supported version by OpenCV. After installation, the following code is configured to extract two random frames from the 12fps .avi video files collected as a dataset.
The results were image files with an aspect ratio of 16:9. The number of sampled images was twice the number of the dataset collected as two random frames were captured from each .avi file. For numberplates, we captured a portion of images from the CCTV footage. As the count of numberplates was inadequate, we manually collected images of numberplates from different locations. However, as the usage of English-embossed number plates was newly introduced in Nepal, we could not gather sufficient numbers. So, we collected the remaining images from different online sources that were available to us.

6.4.4 Dataset Labeling

Once the images were obtained, each cropped images were labeled using labelImg tool. In this software, the images can be annotated easily as it is user-friendly. The software is set to YOLO format and the images were annotated accordingly. The data of the bounding boxes that are annotated are saved in the .txt format which will save the coordinates of the bounding boxes. The whole dataset was labeled into class 'car'.

For numberplates, we used a similar method to assign the 'np' label for traditional numberplates and 'en' for embossed numberplates. For the segmentation model, we changed the labels for the cropped numberplates to the same label 'np' as segmentation required a single class.
For the characters, the same thing as numberplate labeling was done. But, as characters can range from 0-9, A-Z, ba, cha, pa, etc, the labeling task was a lot more extensive. For the segmentation, we did the same thing but changed the labels to 'ch'.

6.4.5 Dataset Integration and Verification

Since the preparation of the dataset and preprocessing task were divided amongst the team members, the dataset was finally compiled into a single directory and the images and their labels were verified for any anomaly.
To train all the models, the datasets had to be divided into training and validation datasets. So we used the traditional Scikit-Learn library to do so. For each of the datasets, 80% of the dataset was for training and the rest 20% for validation.

6.4.6 Model Training

In this project, one model was trained for vehicle detection, and a set of models were trained for ANPDR architecture for numberplate recognition. The number of epochs and batch sizes used were determined and optimized over many iterations according to the performance and training data from various graphs. The best set of parameters were finalized and the models were trained accordingly. The following number of epochs and batch sizes were used for the models:

Table 6.6: Epochs and Batches for Models

Model	**Epochs**	**Batch Size**
Vehicle Detection	150	32
Numberplate Detector	100	16
Numberplate Classifier	100	32
English Character Segmentation	50	16
English Character Recognition	200	16
Nepali Character Segmentation	50	16
Nepali Character Recognition	200	16

6.5 Algorithms and Models

6.5.1 Automatic Numberplate Detection and Recognition

This model is constructed from a set of CNN and YOLO models that is used in detecting numberplates, classifying them into English embossed numberplates and traditional Nepali numberplates. The resultant numberplate segments each character and recognizes it. The following tasks used the following models for their work:

- Numberplate Detector: YOLOv8n architecture
- Numberplate Classifier: YOLOv8n architecture
- English Character Segmentation: YOLOv8n architecture
- English Character Recognition: CNN
- Nepali Character Segmentation: YOLOv8n architecture
- Nepali Character Recognition: CNN

6.5.1.1 Flowchart for Numberplate Detection and Recognition

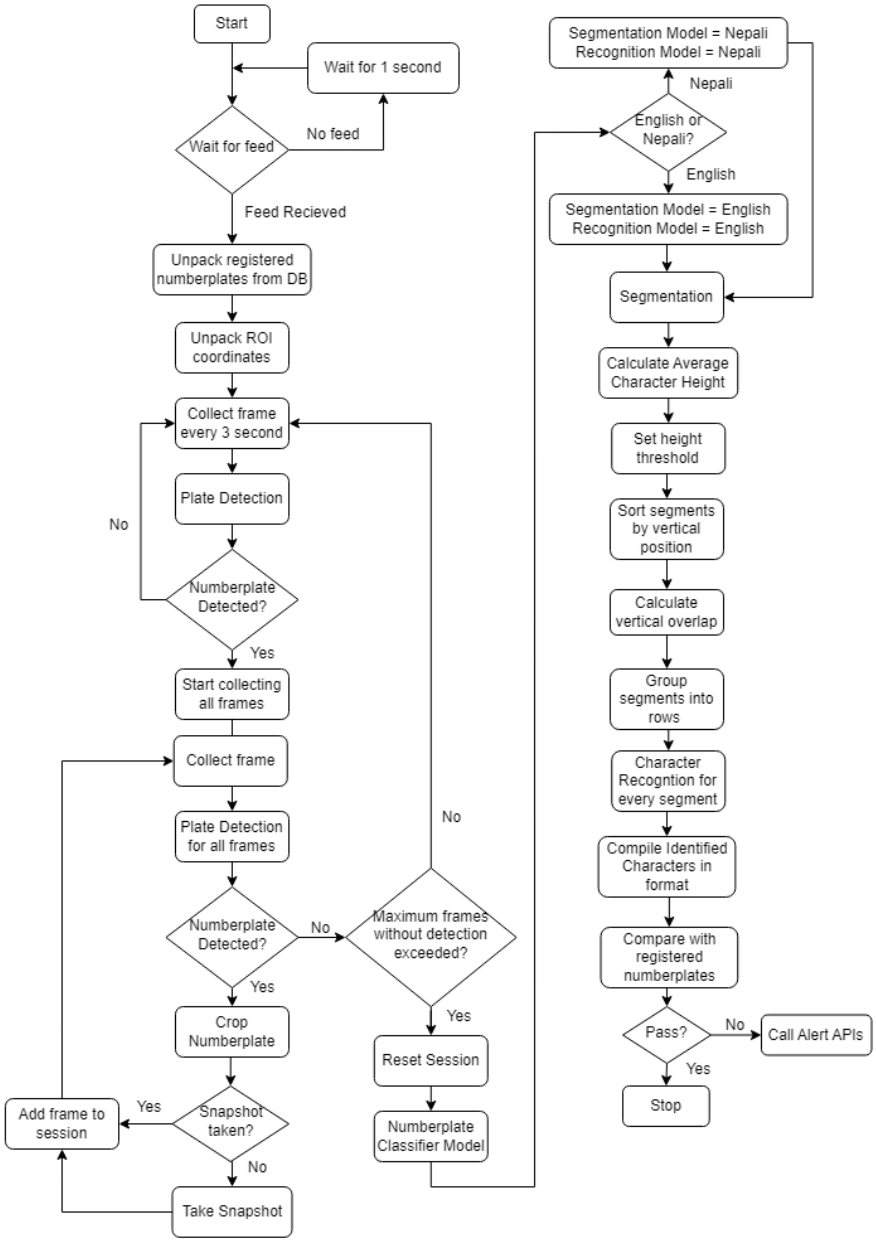


Figure 6.4: ANPDR Flowchart

The steps for the ANPDR Flowchart are described below:

1. In the first step, we wait for the feed of the CCTV camera which is used for the detection of number plates. If the feed is received, the ROI is unpacked. If the feed is not received, the script waits for 1 second and retries again.
2. The program then unpacks the registered numberplates data from database into a temporary list by calling upon an API to the backend server.
3. The next step involves unpacking of ROI coordinates where the numberplate is to be detected.
4. The frames are set to be collected every 3 seconds from the active video feed.
5. The frames are used by the numberplate detection model to check for any numberplates in the scene
6. If a numberplate is detected, the program is now set to collect all frames instead of frames every 3 seconds.
7. A frame is captured and feeded to the plate detection model
8. If a numberplate is detected, the numberplates are cropped from the frames, otherwise it is checked if the maximum number of frames without numberplate detection is reached or not.
9. In the case of numberplate detection, the numberplates are cropped, and a snapshot is taken if not taken previously, and the cropped numberplates are added to the session for further segmentation and classification.
10. If the numberplate detection model does not detect any plates within a certain number of frame limit, the session is reset, and the previous session is sent to the numberplate classification model.
11. The numberplate classification model checks the frames in the session and detects if the numberplate is a Nepali numberplate or an English numberplate. If the plates are Nepali, the segmentation and recognition that are loaded are Nepali models. Otherwise, the English models are loaded into the script.
12. Now, using the loaded model, the segmentation for the group of numberplates is performed and each characters are extracted.
13. The average height of all the characters are calculated.
14. A height threshold is set according to the average height.
15. The vertical overlaping from each segmented characters are calculated.
16. The segments are then grouped into rows according to relative position with the threshold height.
17. Now that the segmentation is performed, each segment's characters are classified using the loaded classification model.
18. The identified characters are then put in an ordered format and compared with the registered number plates stored in the database. If the number plate is matched with the registered number plates, it is set as passed. Otherwise, the alert APIs are called.

NOTES:

- Steps 4 to 6 are being done as to not process every frames when there is no vehicle activity in the region which helps in saving tons of computing power and resource.
- There are many models being used so the computing time for checking the numberplate is usually around 90 ms. As the vehicles in the frame are moving, this amount of processing time leads to skipping of many frames and thus loss of information. Step 7 to 10 solves this problem by first collecting every frame into a list when a numberplate is first detected and counting it as one session. This session ends when a maximum threshold number of frames without detection is exceeded and only then all the frames in the session list are processed by the program.
- Steps 13 to 16 are being done to format the output of the characters in a systematic way of reading them from top left to bottom right. This solution even works when the numberplate images are in an off axis position.

6.6 CNN models

The constructed ANPDR architecture consists of two CNN models :

- Nepali Character Recognition Model
- English Character Recognition Model

6.6.0.1 Nepali Character Recognition Model

The architecture for the Nepali Character Recognition model is shown below.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 batch_normalization (Batch  (None, 62, 62, 32)        128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 31, 31, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        18496

 batch_normalization_1 (Bat  (None, 29, 29, 64)        256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 14, 14, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 12, 12, 128)       73856

 batch_normalization_2 (Bat  (None, 12, 12, 128)       512
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 6, 6, 128)         0
 g2D)

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 256)               1179904

 batch_normalization_3 (Bat  (None, 256)               1024
 chNormalization)

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 batch_normalization_4 (Bat  (None, 128)               512
 chNormalization)

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 13)                1677
```

Figure 6.5: Architecture of Nepali Character Recognition model

Firstly, before training the model, we normalized all the input images. Since the training data was comparatively lower, we had to augment the training data by horizontally flipping, tilting, and zooming the data to increase the dataset. After that, the model was put to train. The model consists of the following layers:

- Conv2D Layers : The input layer of the first Conv2D layer is a 2D grayscale image of size 64x64. The first Conv2D layer applies 32 3x3 spatial windows(filters) which move across the image to produce feature maps. The output of this layer is 32 feature maps of size 62x62. The second Conv2D layer receives the input from the MaxPooling2D layer which this time, uses 64 spatial windows to produce 64 feature maps of size 29x29. The third and final Conv2D layer uses input from the MaxPooling2D 1 layer with 128 filters to produce 128 feature maps of size 12x12. ReLu activation function is applied to each of the outputs.

- Batch Normalization Layers : The model uses 5 Batch Normalization layers. During training, the layer normalizes its output using the mean and standard

deviation of the current batch of inputs. This is done so that the training of the CNN is faster and more stable.

- Max Pooling2D Layers : The model consists of 3 MaxPooling2D layers applied after each pair of Conv2D and BatchNormalization layers. It is a layer similar to the Conv2D layer which uses filters (32 and 64 sizes), but instead of multiplying the input with these filters, the layer takes the maximum value within the window. It is a down-sampling operation that reduces the dimensionality of each feature map while retaining the most important information of the input.

- Flatten Layer : The responsibility of the single Flatten Layer in the network is to convert the output of the MaxPooling2D layer(2D output) into a single-dimensional output which can be provided as input to the Dense layer. The resultant output shape of the Flatten Layer becomes 4608 as the 128 feature maps of 6x6 size get converted into a single dimension (6*6*128 = 4608).

- Dense Layer : The Dense layer is the traditional neural network layer. There are three Dense layers used in the model. The first layer contains 256 neurons while the second and third layer contains 128 and 13 (corresponding to the total number of output classes) respectively. Finally, this layer uses the same ReLu activation functions which are applied to the output of the network except the final layer which uses softmax instead to normalize the output of a network to a probability distribution over predicted output classes. The layer implements the following operation:

$$\text{output} = \text{activation}\left(\text{dot}(\text{input}, \text{kernel}) + \text{bias}\right)$$

- Dropout Layer : There are 2 Dropout Layers in the model. Sometimes due to overfitting, some units may adapt in such a way that it changes the mistakes of other units. So, to avoid this co-adaption problem, we use a Dropout layer that removes these units with a dropout probability of 0.5(set manually). As a result, the model becomes more generalized.

6.6.0.2 English Character Recognition Model

The architecture for the English Character Recognition model is shown below.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        320

 batch_normalization (Batch  (None, 62, 62, 32)        128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 31, 31, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        18496

 batch_normalization_1 (Bat  (None, 29, 29, 64)        256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 14, 14, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 12, 12, 128)       73856

 batch_normalization_2 (Bat  (None, 12, 12, 128)       512
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 6, 6, 128)         0
 g2D)

 flatten (Flatten)           (None, 4608)              0

 dense (Dense)               (None, 256)               1179904

 batch_normalization_3 (Bat  (None, 256)               1024
 chNormalization)

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 batch_normalization_4 (Bat  (None, 128)               512
 chNormalization)

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 36)                4644
```

Figure 6.6: Architecture of English Character Recognition model

The data was first normalized by the same procedure as the Nepali dataset. The training data was adequate and flipping the images could cause ambiguity among the characters in the dataset, so augmentation was not performed. The model consists of the following layers:

- Conv2D Layers : The input layer of the first Conv2D layer is a 2D grayscale image of size 64x64. The first Conv2D layer applies 32 3x3 spatial windows(filters) which move across the image to produce feature maps. The output of this layer is 32 feature maps of size 62x62. The second Conv2D layer receives the input from the MaxPooling2D layer which this time, uses 64 spatial windows to produce 64 feature maps of size 29x29. The third and final Conv2D layer uses input from the MaxPooling2D 1 layer with 128

filters to produce 128 feature maps of size 12x12. ReLu activation function is applied to each of the outputs.

- Batch Normalization Layers : The model uses 5 Batch Normalization layers. During training, the layer normalizes its output using the mean and standard deviation of the current batch of inputs. This is done so that the training of the CNN is faster and more stable.

- Max Pooling2D Layers : The model consists of 3 MaxPooling2D layers applied after each pair of Conv2D and BatchNormalization layers. It is a layer similar to the Conv2D layer which uses filters (32 and 64 sizes), but instead of multiplying the input with these filters, the layer takes the maximum value within the window. It is a down-sampling operation that reduces the dimensionality of each feature map while retaining the most important information of the input.

- Flatten Layer : The responsibility of the single Flatten Layer in the network is to convert the output of the MaxPooling2D layer(2D output) into a single-dimensional output which can be provided as input to the Dense layer. The resultant output shape of the Flatten Layer becomes 4608 as the 128 feature maps of 6x6 size get converted into a single dimension (6*6*128 = 4608).

- Dense Layer : The Dense layer is the traditional neural network layer. There are three Dense layers used in the model. The first layer contains 256 neurons while the second and third layers contain 128 and 36 (corresponding to the total number of output classes) respectively. Finally, this layer uses the same ReLu activation functions which are applied to the output of the network except the final layer which uses softmax instead to normalize the output of a network to a probability distribution over predicted output classes. The layer implements the following operation:

$$\text{output} = \text{activation}\left(\text{dot}(\text{input}, \text{kernel}) + \text{bias}\right)$$

- Dropout Layer : There are 2 Dropout Layers in the model. Sometimes due to overfitting, some units may adapt in such a way that it changes the mistakes of other units. So, to avoid this co-adaption problem, we use a Dropout layer that removes these units with a dropout probability of 0.5(set manually). As a result, the model becomes more generalized.

6.7 Parking Space Availability Detection

This model consists of a model trained on Yolov8n that was used to detect and recognize available parking spaces from unrecognized parking spaces. This was used to update our React App Overview of the parking space.

6.7.1 Flowchart for Vehicle Detection

The steps in the Vehicle Detection Flowchart are described below:

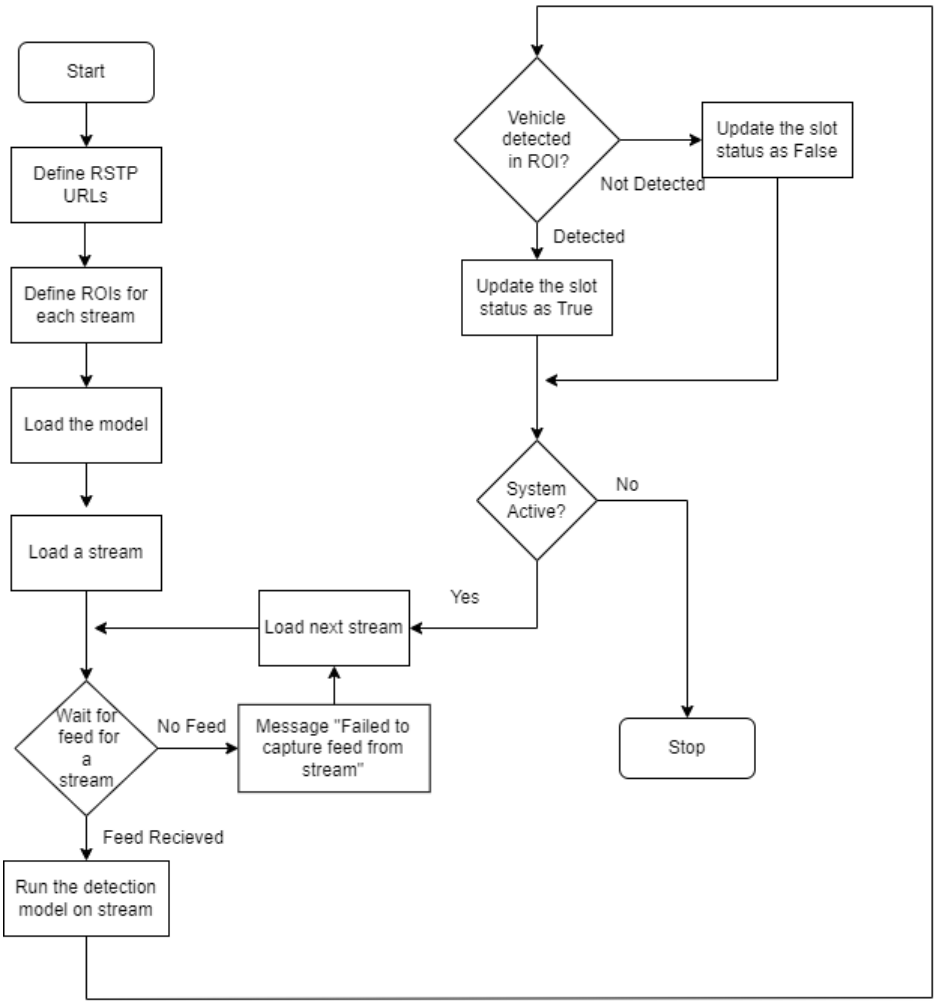


Figure 6.7: Vehicle Detection Flowchart

1. Each CCTV is loaded into the model using Real-time Streaming Protocol(RTSP). So, in the first step, RTSP URLs are defined for each CCTV camera in a single list.
2. The second step involves defining the Region of Interest (ROI) for each stream. The ROI defines the coordinates of where each vehicle should be in its slot.
3. The third step is to load the model into the script. For vehicle detection, the vehicle detection model is loaded into the script.
4. A single URL is selected from the list and the stream is loaded in the script.
5. The script waits for the stream. If the feed is received, the previously loaded model is run on the stream. Otherwise, a message is sent indicating the failure to capture feed from the stream and, the next stream from the list is loaded and the step is repeated.
6. Once the feed is loaded, the detection model runs for the stream where the vehicle detection process takes place.
7. The model checks if any vehicle is detected on the ROI of the stream. If any vehicle is detected, the slot status that defines the parking slot is set as

True, otherwise, the status of the slot is set as False.

8. The next step involves checking if the system is still active to continue the ongoing vehicle detection process. If the system is inactive, the script is stopped, otherwise, the next stream is loaded and the vehicle detection model continues to detect vehicle occupancy in the parking slots.

6.8 YOLOv8n

YOLOv8 provides five different models based on size and performance. Among these models, the architecture of the model that was used in the project was YOLOv8n. The reasons for choosing YOLOv8n were:

- The model provided high efficiency and had higher detection speed making it suitable for the system.
- As there was the use of several models in the system, there was the necessity to use lightweight models for which YOLOv8n was a great option.
- Despite YOLOv8n delivering limited accuracy, the accuracy required for the system was sufficient using the model.
- Due to adequate online information about the deployment, the model was easy to use and deploy.
- The system was cost-effective after using the lightweight model, hence reducing the economic and computational costs to run the system.
- As the system required edge computing, due to its lightweight feature, the model could be run residing by the location of the parking space, which is limited in resources.

6.8.1 Mosaic Augmentation

Mosaic data augmentation is a simple augmentation technique in which four different images are stitched together and fed into the model as input. This makes the model learn the actual objects from different positions and in partial occlusion.

Figure 6.8: Mosaic Augmentation

6.9 Activation Functions

6.9.1 Rectified Linear Unit(ReLU) Activation Function

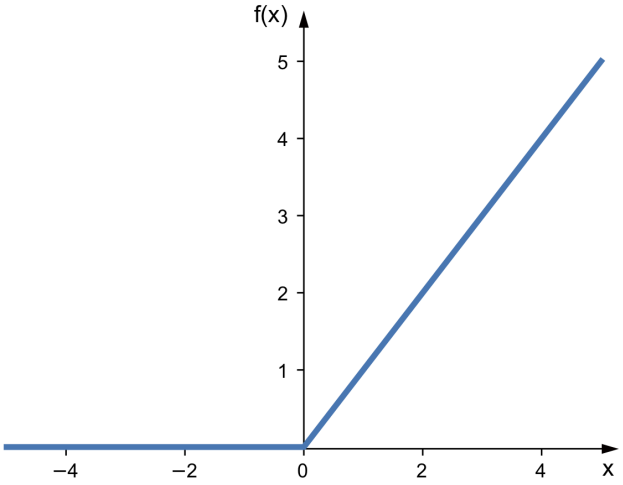


Figure 6.9: ReLU activation function

Rectified Linear Unit (ReLU) is a commonly used activation function in Convolutional Neural Networks (CNNs) due to its simplicity and effectiveness. ReLU introduces non-linearity by outputting zero for negative input values and retaining positive input values unchanged. This non-linear behavior allows CNNs to learn complex patterns and relationships within data, facilitating better feature representation and improving the model's ability to capture intricate patterns in images. Additionally, ReLU helps mitigate the vanishing gradient problem, which can occur during the training process, by allowing for faster convergence and reducing the likelihood of saturation in the activation layers. Overall, ReLU's computational efficiency, simplicity, and ability to address the limitations of earlier activation functions make it a popular choice in CNN architectures, contributing to improved performance and faster training times. The function of ReLU is given as:

$$f(x) = \max(0, x)$$

6.9.2 Softmax Activation Function

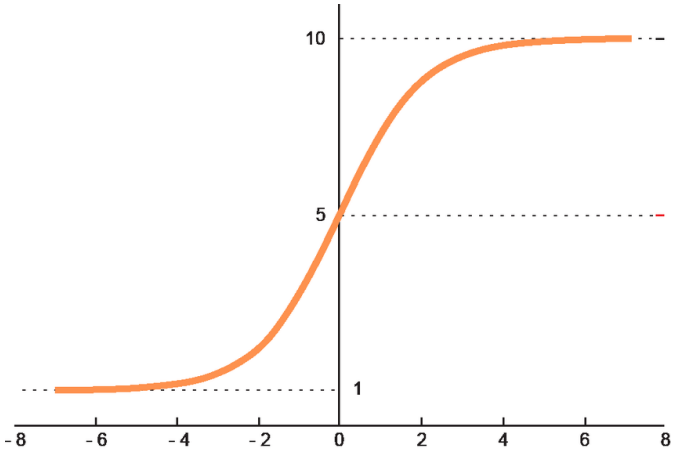


Figure 6.10: Softmax activation function

Softmax is a mathematical function that converts a vector of arbitrary real-valued scores into a probability distribution. It is commonly used in machine learning and deep learning tasks, particularly in classification problems. The softmax function takes as input a vector of scores or logits and transforms them into probabilities such that each score is mapped to a value between 0 and 1, and the sum of all probabilities is equal to 1. This makes softmax suitable for multiclass classification tasks where the model needs to output probabilities for each class. In convolutional neural networks (CNNs), softmax is often used in the output layer to generate class probabilities from the final feature representation obtained after multiple layers of convolutions, pooling, and fully connected layers. By using softmax, CNNs can effectively classify input images into different categories with a probabilistic interpretation, making them suitable for the developed CNN. The function of Softmax is given as:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

6.9.3 Mish Activation Function

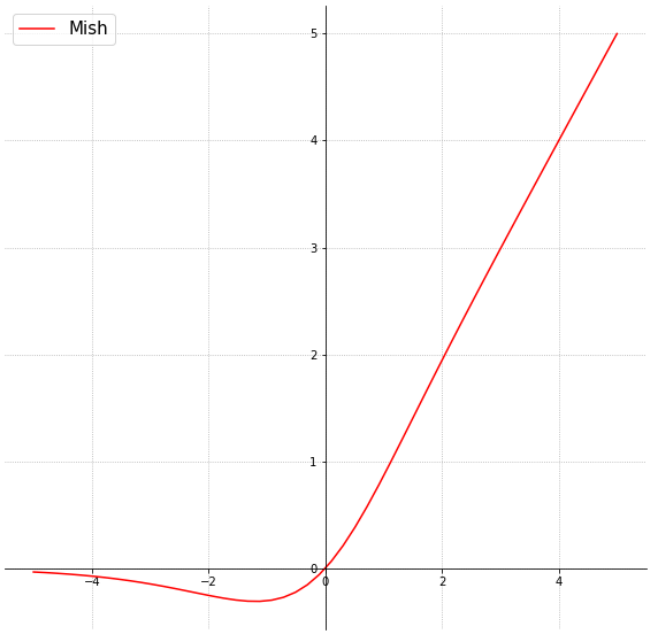


Figure 6.11: Mish activation function

Mish activation function is the default activation function of YOLOv8 architecture defined by the mathematical expression given below:

$$\text{Mish}(x) = x \cdot \tanh(\text{softplus}(x))$$

$$\text{where softplus}(x) = \ln(1 + e^x)$$

ReLU activation function shows a dying ReLU phenomenon, which is gradient information loss due to the reason negative inputs are outputted to zero. So, to eliminate the dying ReLU problem, the activation function of YOLOv8 was kept as Mish instead of migrating to a simpler ReLU function. The Mish activation function preserves a small amount of negative weights, unlike ReLU which shows better accuracy and improvement in the outputs.

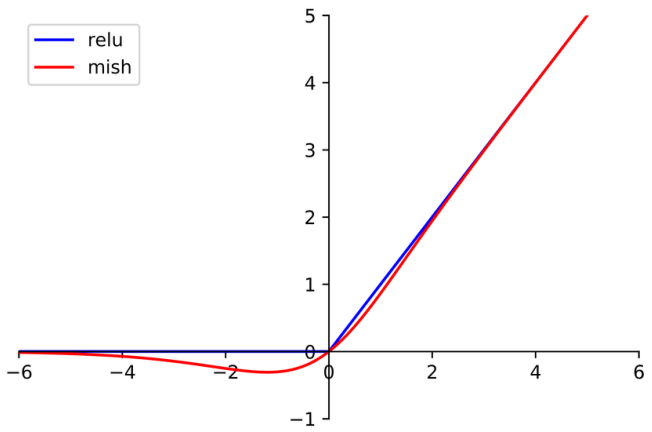


Figure 6.12: Comparison of Mish to ReLU activation function

6.10 Loss Function

6.10.1 Categorical Cross Entropy

Categorical cross-entropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one. Formally, it is designed to quantify the difference between two probability distributions.

$$\text{CategoricalCrossentropy}(y, \hat{y}) = -\sum_i y_i \cdot \log(\hat{y}_i)$$

where $\hat{y}_i$ is the i-th scalar value in the model output, and y_i is the corresponding target value in the model output.

This loss is a very good measure of how distinguishable two discrete probability distributions are from each other. In this context, y_i is the probability that event i occurs and the sum of all y_i is 1, meaning that exactly one event may occur. The minus sign ensures that the loss gets smaller when the distributions get closer to each other.

6.11 Optimizer

6.11.1 Adam Optimizer

Adaptive Moment Estimation is an algorithm for optimization techniques for gradient descent. The method is efficient when working with large problems involving a lot of data or parameters. It requires less memory and is efficient. Intuitively, it is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm.

Adam Optimizer involves a combination of two gradient descent methodologies:

Momentum
This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the 'exponentially weighted average' of the gradients. Using averages makes the algorithm converge toward the minimum at a faster pace.

$$v_t = \beta_1 v_{t-1} + (1 - \beta_1) g_t$$

where:

v_t = momentum term at time step t

β_1 = exponential decay rate for the first moment estimate (typically set to 0.9)

v_{t-1} = momentum term at the previous time step

g_t = gradient at time step t

Root Mean Square Propagation
Root mean square propagation or RMSprop is an adaptive learning algorithm that

tries to improve AdaGrad. Instead of taking the cumulative sum of squared gradients like in AdaGrad, it takes the 'exponential moving average'. Adam Optimizer inherits the strengths or the positive attributes of the above two methods and builds upon them to give a more optimized gradient descent.

6.12 Model Evaluation

Evaluation for the output of each model is done by analyzing the precision, recall and mAP values of each of the implemented models.

6.12.1 Precision

Precision measures the accuracy of positive predictions, i.e., the fraction of correctly predicted positive examples out of all positive predictions made by the model. Here, precision is the number of prediction characters that correctly match actual characters, divided by the total number of predictions in the particular class. It is calculated as:

$$\text{Precision} = \frac{\text{Number of true positive predictions}}{\text{Number of all positive predictions}}$$

6.12.2 Recall

Recall is a metric that quantifies the number of correct positive predictions made out of all positive predictions that could have been made. Here, recall is the number of prediction characters that correctly match actual characters, divided by the total number of actual characters in the particular class. It is calculated as:

$$\text{Recall} = \frac{\text{Number of true positive predictions}}{\text{Number of all actual positive examples}}$$

6.12.3 mAP50

mAP50 is a measure that takes into account both precision and recall at an IoU threshold of 50%. The area under the precision-recall curve gives us mAP. It's a measure of the model's accuracy considering only the "easy" detections.

6.12.4 mAP50-95

mAP50-95 is the metric that takes into account both precision and recall at varying IOU thresholds from 50% to 95%. Unlike mAP50, it gives a comprehensive view of the model's performance across different levels of detection difficulty.

6.12.5 Confusion Matrix

A confusion matrix is a table that is used to define the performance of a classification algorithm. A confusion matrix visualizes and summarizes the performance of a classification algorithm.

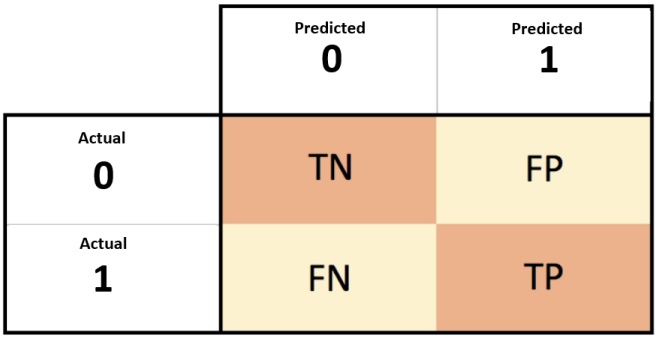


Figure 6.13: Confusion Matrix Diagram

6.13 Test Case

For evaluating the ANPDR system, we designed a set of test datasets, captured from CCTV footage as well as manually collecting the numberplates on our own. The test datasets contain an equal proportion of English-embossed numberplates and Nepali traditional numberplates. These test datasets were provided as inputs to existing OCR models as well, to compare the ANPDR system with the OCR models. The following table shows the number of datasets designed as test cases:

Table 6.7: Test Case Quantity

Type of dataset	**Quantity**
English embossed	100
Nepali traditional	100

The characteristics of the collected test case datasets were:

- The images collected were under different lighting conditions like daytime, where the numberplate shone in the sunlight, during nighttime when the numberplates were filled with noise. Some images were taken under normal lighting conditions as well.
- The images collected were at different angle perspectives, to test the model's working capacity.

6.14 Mobile Application Development

In the development of our application, a modern and robust architecture was implemented, leveraging the strengths of both React for the front end and Java for the back end, seamlessly integrated through the use of various APIs.
The architectural decisions made in the development of our application - from choosing React for its efficient and dynamic user interface capabilities to leveraging Java's robustness and security for the back end, and the strategic use of APIs for integration - have all contributed to creating a scalable, secure, and user-friendly application. These choices reflect our commitment to leveraging cutting-edge technology to provide the best possible experience for our users.

6.14.1 Front-End Development

React, a popular JavaScript library for building user interfaces was chosen for front-end development due to its component-based structure, which allows for the creation of dynamic and responsive web applications. This choice facilitated the development of a highly interactive and user-friendly interface, enabling users to experience smooth navigation and real-time updates without the need for page reloads.

6.14.2 Back-End Development

On the server side, Java was selected for its strong performance, security features, and portability across platforms. By utilizing Java for the back end, we were able to create a robust and scalable server infrastructure capable of handling high volumes of requests efficiently. This choice ensures that our application can grow and adapt to increasing user demands without compromising on performance or security.

6.14.3 API Development

The connection between the front-end and back-end components of our application was meticulously designed using various APIs, enabling seamless communication and data exchange. These APIs were carefully crafted to ensure that data is transferred securely and efficiently, allowing for real-time interactions and updates to be reflected across the user interface with minimal latency. The use of APIs not only facilitated the integration of the front-end and back-end but also provided a flexible foundation for future expansions, such as integrating third-party services or adding new features to our application.

6.15 Server Deployment

In this section of our report, we detail the sophisticated infrastructure established to support and enhance the functionality of our application. Our deployment strategy is meticulously designed to incorporate both cloud and on-premises components, specifically tailored to meet the demands of our application's backend and specialized AI-driven tasks.

By leveraging a dual-server deployment strategy, combining the global reach and scalability of AWS with the specialized processing capabilities of a dedicated home server, our application architecture achieves a harmonious balance between general backend services and intensive AI-driven tasks. This hybrid approach not only enhances the application's functionality, particularly in handling real-time surveillance data, but also ensures a secure, scalable, and cost-efficient infrastructure. The deployment model we have adopted allows us to provide advanced features, such as live surveillance feed analysis, with high performance and reliability, thus meeting the sophisticated needs of our users.

6.15.1 Cloud Hosting with AWS

For the backend of our application, we have chosen Amazon Web Services (AWS) for its robust, scalable, and secure cloud hosting capabilities. The decision to utilize AWS as our primary hosting solution is driven by its extensive suite of services, reliability, and ease of integration with other cloud services. Our backend is deployed on an AWS environment, specifically configured to handle web requests and serve as the central node for application data processing and storage.

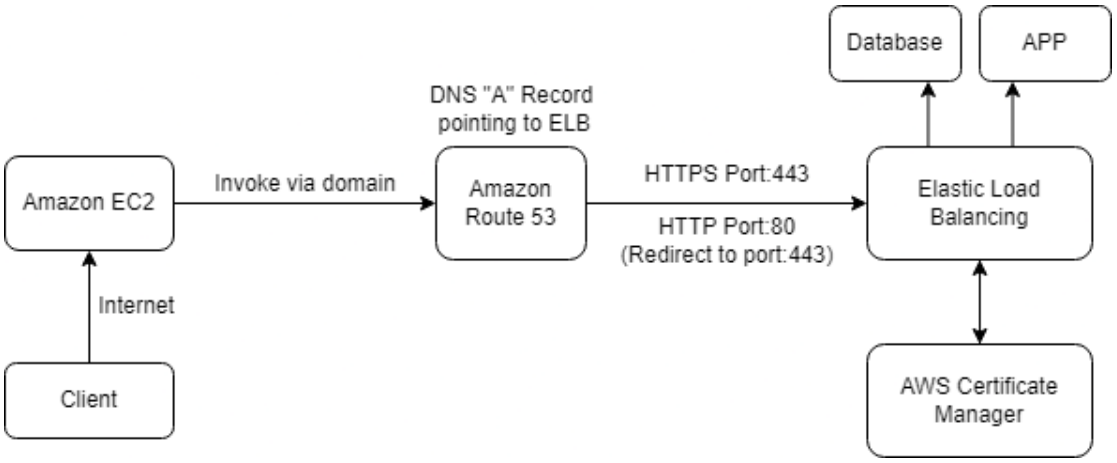


Figure 6.14: AWS network map

This backend is directly linked to a registered domain name, enhancing the accessibility and professionalism of our application. Moreover, we have implemented HTTPS protocol to ensure secure communication between the client and server. This not only protects our user's data but also boosts the credibility and search engine ranking of our application. By leveraging AWS's infrastructure, we ensure that our application is both scalable and resilient, capable of handling varying loads with minimal latency.

6.15.2 Dedicated Home Server for AI and Surveillance Processing

A crucial component of our deployment architecture is the dedicated home server, running on a Linux-based system. This server is uniquely configured to manage real-time CCTV camera feeds, performing sophisticated image processing using trained models for detection and comparison tasks. This setup is instrumental in enabling advanced surveillance capabilities within our application, such as real-time monitoring, anomaly detection, and other AI-driven functionalities.

The integration between our dedicated home server and the AWS-hosted backend is achieved through secure APIs, ensuring efficient and safe data exchange. This configuration allows our application to offload intensive AI processing tasks from the cloud, optimizing overall performance and cost. The home server's role is particularly vital for processing live video feeds, where immediate data analysis is required to provide actionable insights or alerts within the application.

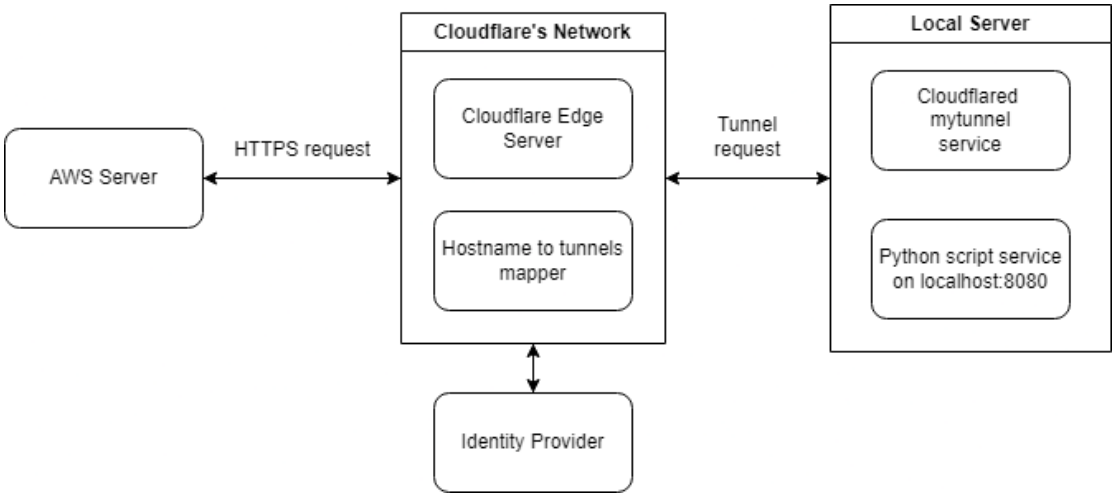


Figure 6.15: Homeserver network map

Due to NAT route limitations by the ISPs, a local tunnel was established for securing connections between the home server with the cloud server. This was achieved with a tunnel service running on the local device that routinely maps the NAT IP address and establishes a connection with the Cloudflare edge server. Here the Domain name is mapped to the tunnel instance used by the homeserver. This makes it possible for a secure HTTPS connection to be established to the server bypassing the NAT restrictions.

Chapter 7

Results and Discussion

7.1 Model Evaluation

7.1.1 Unit Testing

All the models that were trained except the vehicle detection model, were integrated into the ANPDR system. So, we performed unit testing for each of the models before they were system tested. For the CNN models, we constructed Accuracy vs. epoch and Loss vs. epoch for both training and validation datasets. We also created a Confusion Matrix for these models on the test dataset. For the YOLOv8n models, we checked the precision, recall, mAP50, and mAP50-95 graphs. We also constructed the Confusion Matrices for these models. The resultant graphs and matrices for each of the models are shown below.

7.1.1.1 Vehicle Detection Model

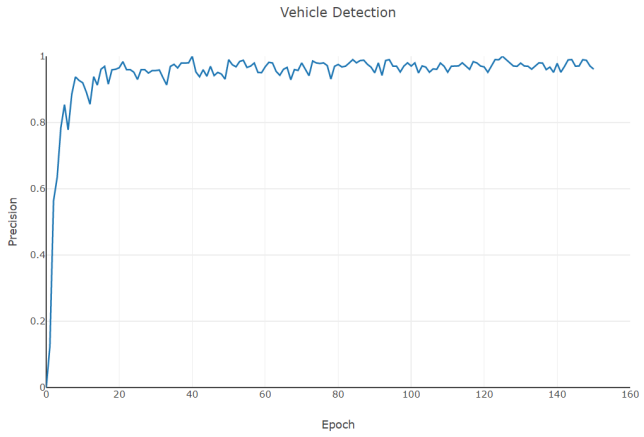


Figure 7.1: Precision Graph of Vehicle Detection

The graph presents the precision metric over epochs for a vehicle detection model. The precision quickly reaches and maintains a level close to 1.0 after an initial increase, which suggests that the model is accurate in its predictions, with a high ratio of true positive detections relative to the total number of positive detections (true and false positives). The precision remains fairly consistent throughout the training, albeit with slight fluctuations which are common in model training. The sustained high precision indicates the model's robustness in accurately detecting vehicles.

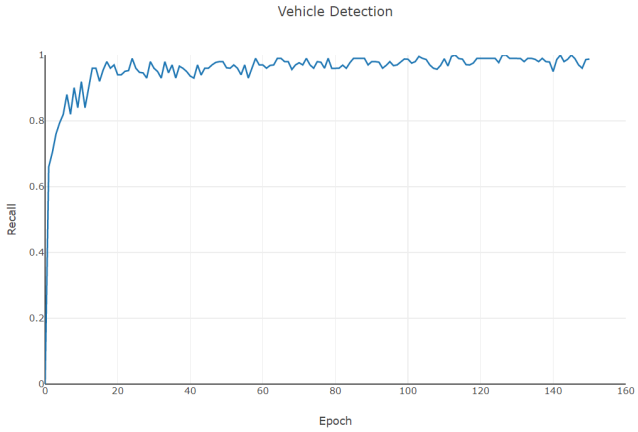


Figure 7.2: Recall Graph of Vehicle Detection

The graph represents the recall metric over epochs for a vehicle detection model. Recall measures the model's ability to correctly identify all relevant instances. The recall quickly ascends to a high level close to 1.0, indicating that from the beginning of the training, the model is able to identify nearly all actual positive instances. Throughout the training process, recall remains consistently high, with only minor fluctuations, demonstrating the model's stable and robust ability to detect vehicles without missing many true positives. This sustained high level of recall across the epochs suggests that the model is reliably capturing the relevant objects it was trained to detect.

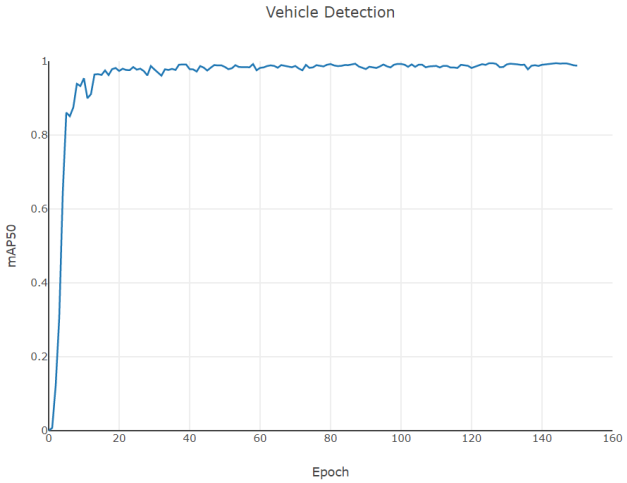


Figure 7.3: mAP50 Graph of Vehicle Detection

The graph shows the mean Average Precision at 50% threshold (mAP50) for a vehicle detection task over approximately 160 epochs. The mAP50 rapidly increases to a high level, almost reaching 1.0, in the initial epochs, suggesting that the model quickly becomes proficient at accurately detecting vehicles. After this initial increase, the mAP50 remains relatively stable with minor fluctuations throughout the training process, indicating consistent and reliable detection performance. The sustained high mAP50 value over the epochs implies that the model maintains its accuracy over time, effectively detecting vehicles within the given data.

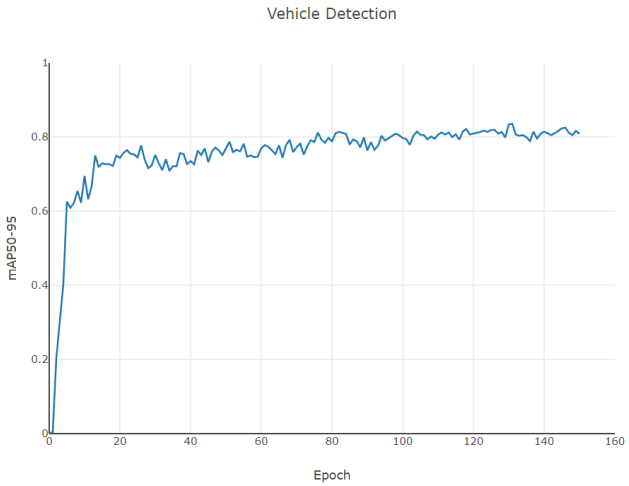


Figure 7.4: mAP50:95 Graph of Vehicle Detection

The graph illustrates the mean Average Precision at thresholds ranging from 50% to 95% (mAP50:95) for a vehicle detection task over approximately 160 epochs. The mAP50:95 climbs sharply at the beginning, indicating that the model quickly learned to detect vehicles with high precision at a range of Intersection over Union (IoU) thresholds. After the initial increase, the mAP50:95 fluctuates slightly but generally maintains a level close to 0.8. This shows that the model has a strong and stable detection performance across varying degrees of IoU thresholds, which suggests good generalization across different detection criteria. The small fluctuations are typical in training dynamics and do not significantly detract from the overall high performance of the model.

7.1.1.2 Numberplate Detection Model

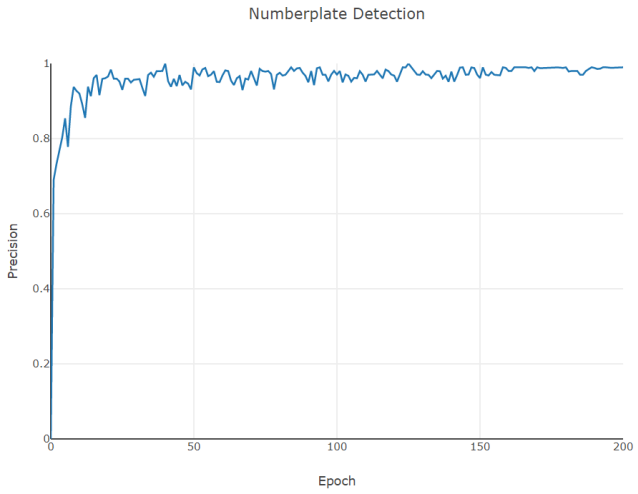


Figure 7.5: Precision Graph of Numberplate Detection

The precision graph for number plate detection over 200 epochs indicates a rapid increase in high levels of precision within the first few epochs. The model achieves and maintains precision above 0.8, demonstrating its ability to correctly identify

true positive detections while minimizing false positives. The precision remains relatively stable throughout the training period, with only minor fluctuations, suggesting that the model is robust in its detection capabilities and maintains this performance consistently over time.

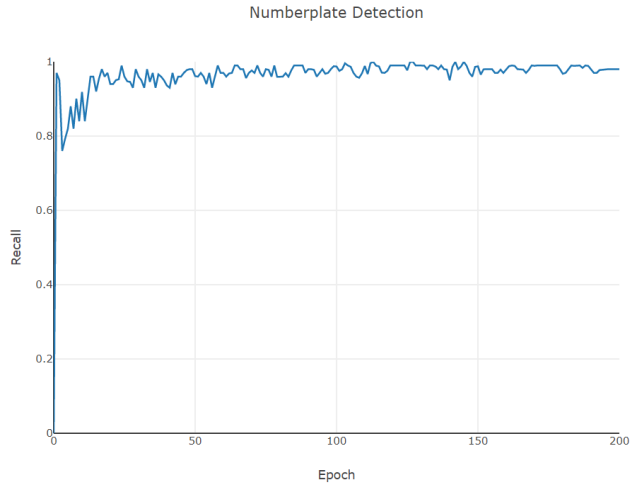


Figure 7.6: Recall Graph of Numberplate Detection

The recall graph for number plate detection over 200 epochs shows a swift climb to high levels of recall, stabilizing just below 1.0. This metric indicates the model's success in identifying most of the relevant instances (true positives) across the training period. The recall remains consistently high with very slight fluctuations, demonstrating that the model reliably detects number plates, with few instances missed (low false negatives). The stability of recall at such a high level suggests that the model has a strong capability to find and correctly label number plates in the data it was tested on.

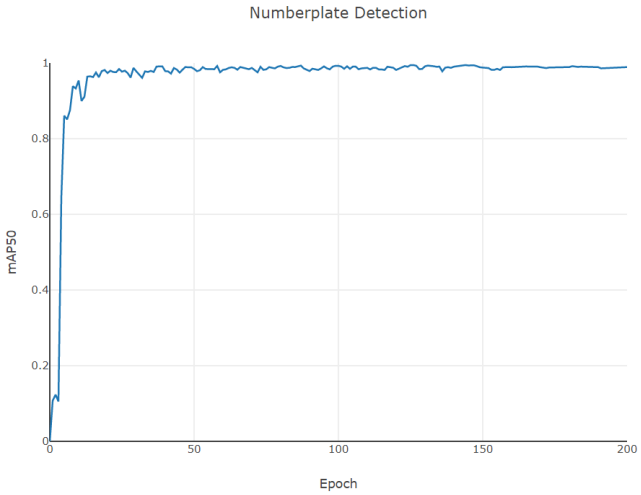


Figure 7.7: mAP50 Graph of Numberplate Detection

The graph shows the mean Average Precision at 50% threshold (mAP50) for a number plate detection task over 200 epochs. The mAP50 value quickly increases at the start of the training and reaches a high level near 1.0, indicating the model's

strong capability in accurately detecting number plates. After the initial rise, the mAP50 shows very little variation and maintains a high value, which suggests that the model consistently performs well in detecting number plates with a high precision across the span of training epochs. The slight fluctuations seen are typical in the training of deep learning models and do not significantly affect the overall high detection performance.

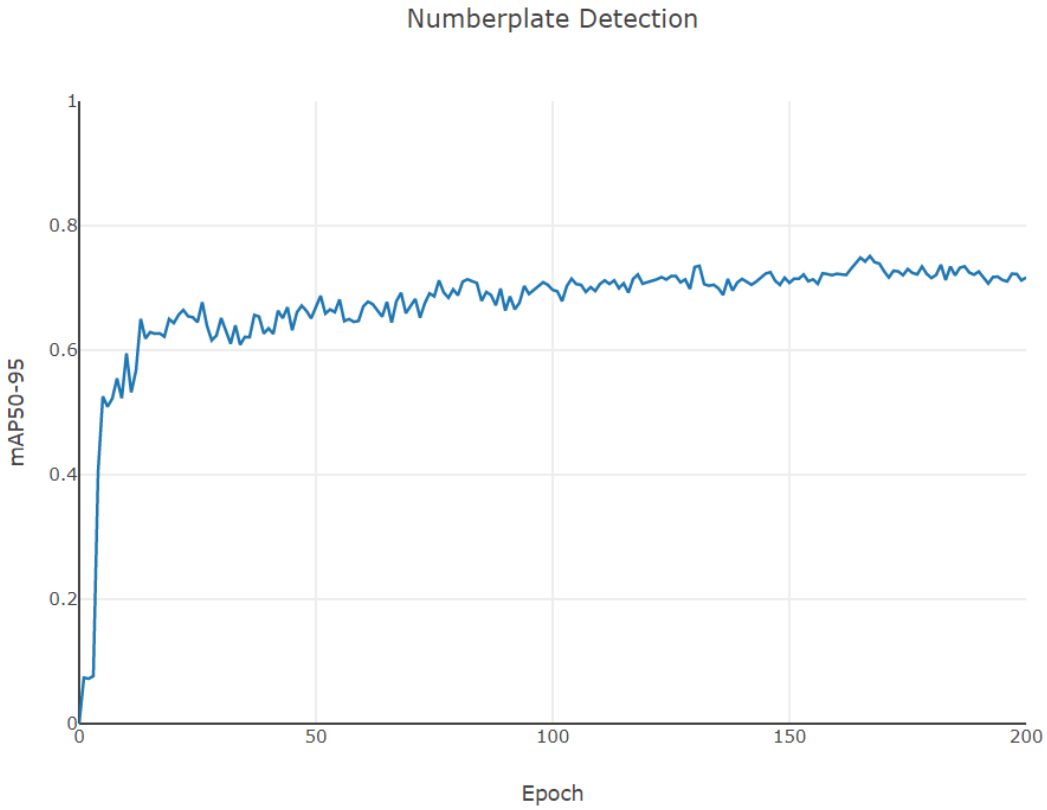


Figure 7.8: mAP50:95 Graph of Numberplate Detection

The graph displays the mean Average Precision at thresholds ranging from 50% to 95% (mAP50:95) for number plate detection over 200 epochs. The mAP50:95 rises sharply early on, suggesting quick learning and improvement in detecting number plates at various Intersections over Union (IoU) thresholds. The metric levels off after the initial rise, indicating that the model maintains a relatively consistent performance throughout the remainder of the training. The overall trend is a gradual increase, with some fluctuations, which is typical in a learning process as the model adjusts to better fit the data. The graph shows that the model has achieved and sustained a good level of precision in detecting number plates across more strict IoU thresholds.

7.1.1.3 Numberplate Classification Model

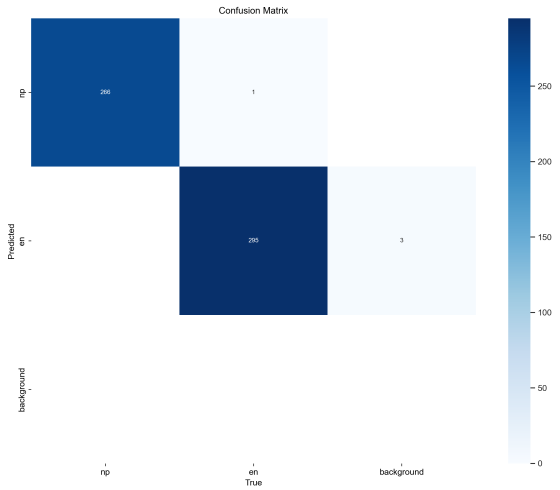


Figure 7.9: Confusion Matrix of Numberplate Classification

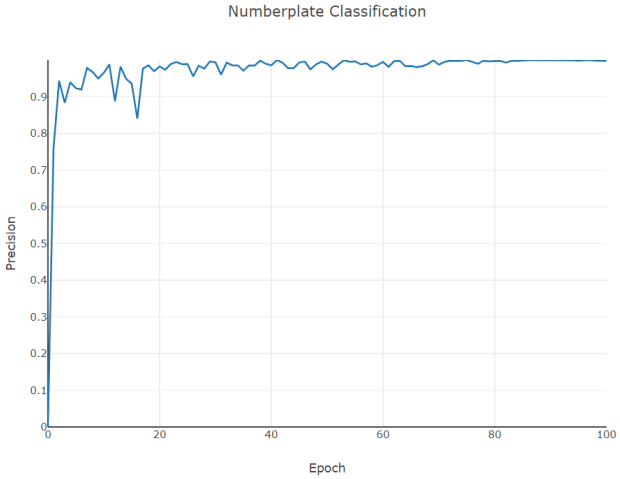


Figure 7.10: Precision Graph of Numberplate Classification

The precision graph for number plate classification over 100 epochs shows a swift ascent to a high level of precision, reaching above 0.9 within the initial epochs. Following this rapid increase, the precision levels out, maintaining a high value with slight fluctuations throughout the remaining epochs. This demonstrates that the model is consistently making accurate positive predictions (true positives) while minimizing the false positives, which is indicative of a highly precise model for this classification task. The minor variations observed are typical in model training and do not significantly detract from the overall high performance.

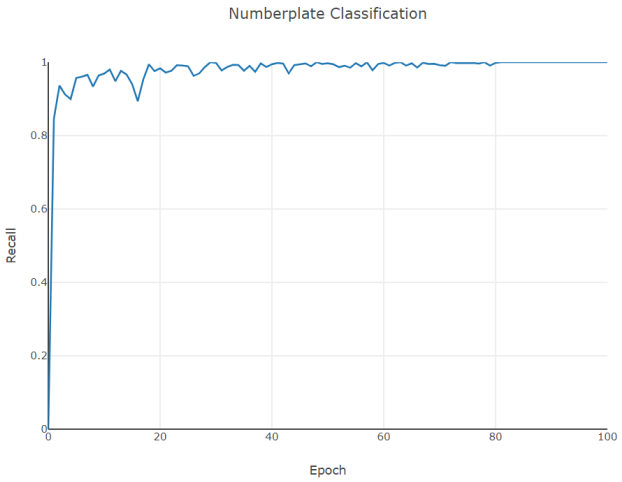


Figure 7.11: Recall Graph of Numberplate Classification

The graph for the recall metric in number plate classification over 100 epochs indicates that the model quickly achieves a high recall rate, close to 1.0, early in the training process. This high recall is sustained with minor fluctuations throughout the training, showing that the model consistently identifies the true positive number plates. The high and stable recall suggests that the model successfully minimizes false negatives, meaning it rarely misses a number plate that should have been classified or detected. This is indicative of a model with a strong ability to detect all relevant instances within the dataset across the duration of its training.

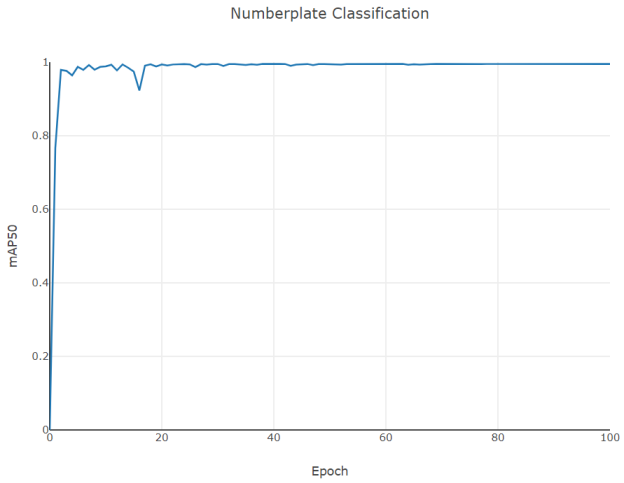


Figure 7.12: mAP50 Graph of Numberplate Classification

The graph shows the mean Average Precision at 50% threshold (mAP50) for a number plate detection task over 200 epochs. The mAP50 value quickly increases at the start of the training and reaches a high level near 1.0, indicating the model's strong capability in accurately detecting number plates. After the initial rise, the mAP50 shows very little variation and maintains a high value, which suggests that the model consistently performs well in detecting number plates with a high precision across the span of training epochs. The slight fluctuations seen are

typical in the training of deep learning models and do not significantly affect the overall high detection performance.

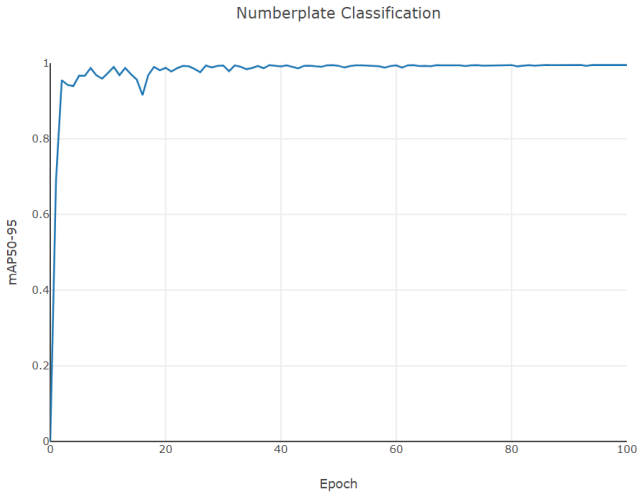


Figure 7.13: mAP50-95 Graph of Numberplate Classification

The graph shows the mean Average Precision at thresholds ranging from 50% to 95% (mAP50:95) for a number plate classification task over 100 epochs. The mAP50:95 metric shows a steep increase early in the training, reaching above 0.8, which suggests that the model quickly became adept at classifying number plates across a range of intersections over union (IoU) thresholds. After this initial rapid improvement, the metric plateaus, indicating that further epochs did not result in significant performance gains. The relatively flat line, with minor fluctuations, reflects a stable and consistent model performance across the more stringent IoU thresholds. This high and stable mAP50:95 value suggests that the model is robust and reliable in its number plate classification task across a variety of conditions.

7.1.1.4 Nepali Numberplate Segmentation Model

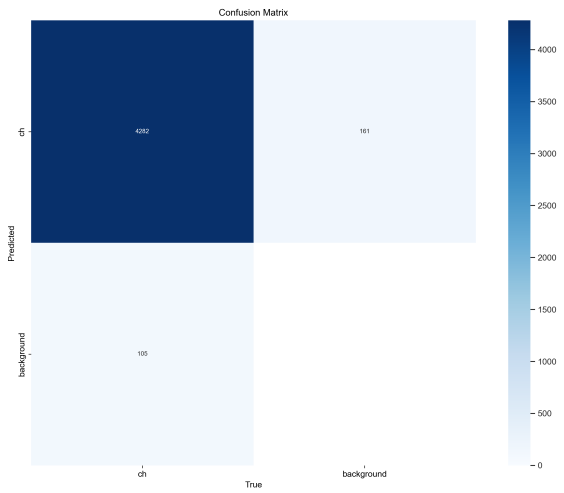


Figure 7.14: Confusion Matrix of Nepali Character Segmentation

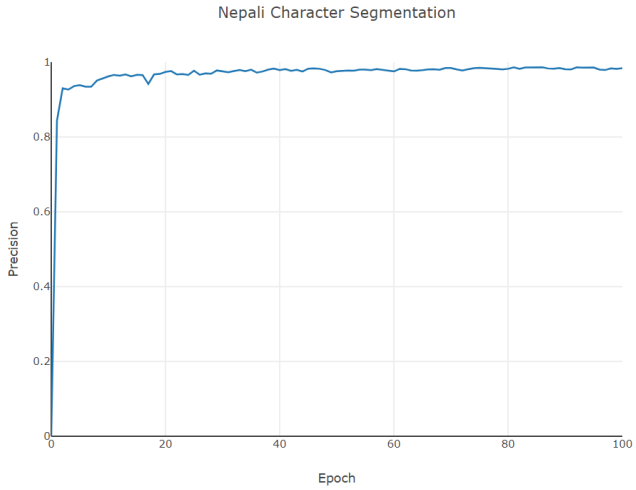


Figure 7.15: Precision Graph of Nepali Character Segmentation

The graph shows the precision metric for a Nepali character segmentation task over 100 epochs. The precision quickly reaches a value close to 1.0, indicating that the model is highly accurate in predicting true positives right from the early epochs. After the initial rise, the precision levels off, showing very little variance and maintaining a high level throughout the remainder of the epochs. This indicates a stable and reliable performance from the model in terms of precision, with consistently few false positives being predicted as the model continues to learn and validate its predictions.

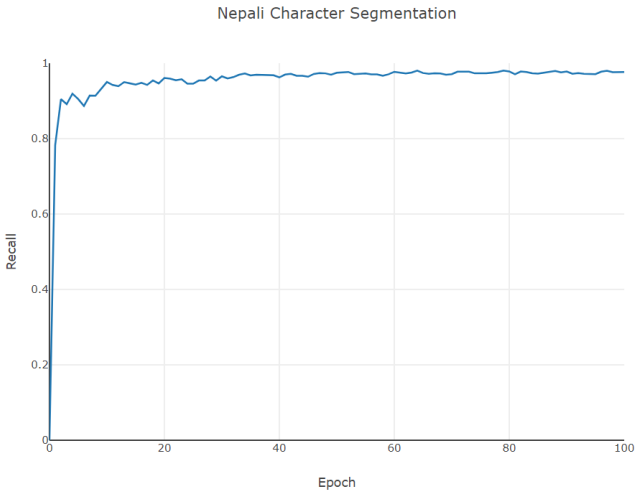


Figure 7.16: Recall Graph of Nepali Character Segmentation

The graph illustrates the recall metric for Nepali character segmentation throughout 100 epochs. Recall measures the ability of the model to identify all relevant instances. In this graph, recall quickly ascends to a high value near 1.0, indicating that the model is successfully identifying most of the true positive cases from the very beginning. The metric remains consistently high with minimal fluctuation, which signifies that the model maintains its ability to capture the majority of positive examples throughout the training process without missing many actual cases.

This level of recall suggests the model has strong sensitivity for the segmentation task at hand.

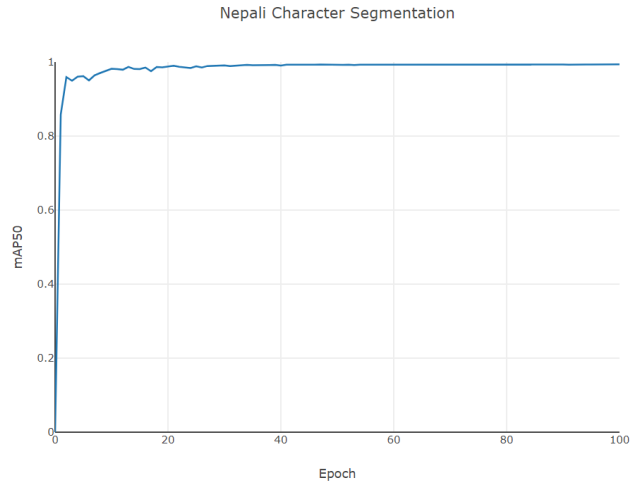


Figure 7.17: mAP50 Graph of Nepali Character Segmentation

The graph indicates the mean Average Precision at 50% threshold (mAP50) for Nepali character segmentation across 100 epochs. The mAP50 value rises very quickly to a high level near 1.0 within the initial few epochs. It then remains consistently high throughout the rest of the training process, with very little variation. This demonstrates that the model achieved a strong segmentation performance almost immediately and maintained that level of performance, showing a high degree of precision in detecting and segmenting the correct characters from the very beginning of its training. The flatness of the line after the initial rise suggests that the model was able to learn the segmentation task quickly and did not require significant further adjustments to maintain high precision.

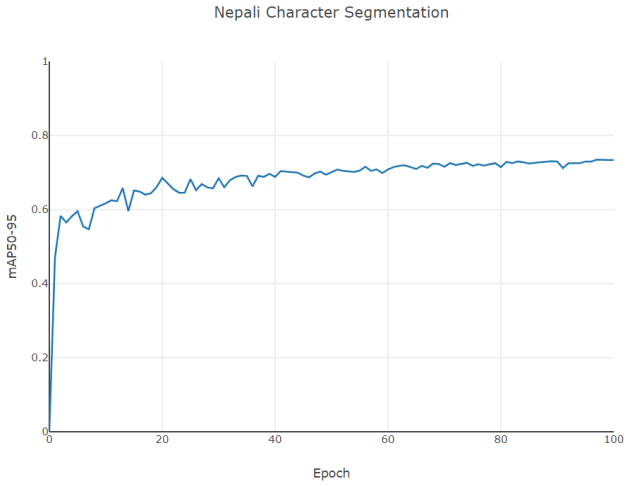


Figure 7.18: mAP50:95 Graph of Nepali Character Segmentation

The graph displays the performance of a model on Nepali character segmentation over 100 epochs, measured by mean Average Precision at thresholds ranging from 50% to 95% (mAP50:95). The mAP50:95 metric shows a rapid increase during the early epochs, reaching over 0.6, and then continues to climb at a more

gradual rate. This progression suggests that the model is steadily improving at segmenting characters with stricter criteria for precision. The metric experiences some fluctuations but maintains a generally upward trend, indicating that while the model is performing well, there is room for improvement in consistency. The overall high level of mAP50:95 maintained from early on through to epoch 100 indicates robust segmentation capabilities, but the fluctuations could suggest that the model might benefit from further refinement to stabilize its performance at the stricter thresholds.

7.1.1.5 English Numberplate Segmentation Model

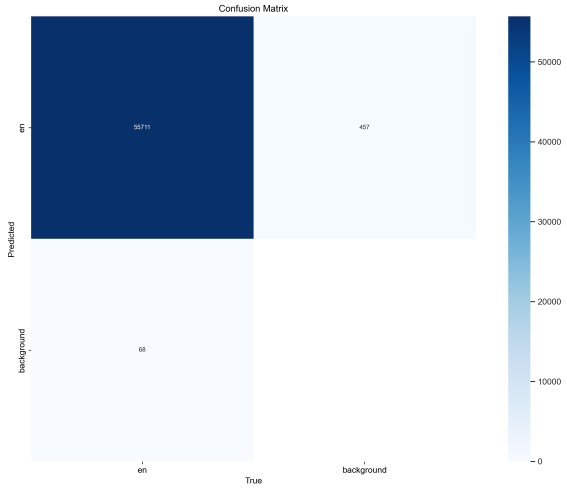


Figure 7.19: Confusion matrix of English Character Segmentation

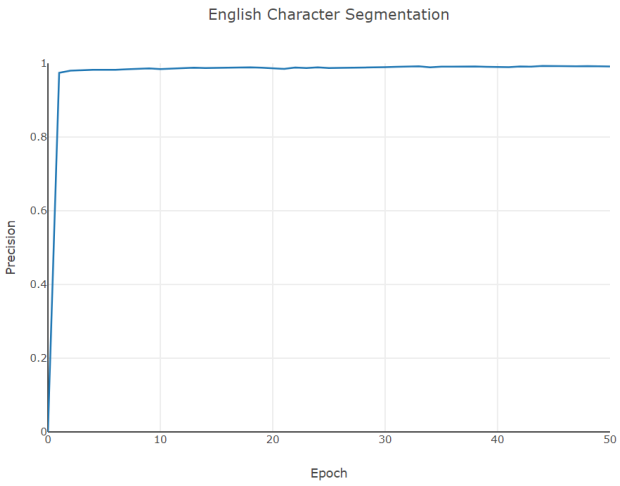


Figure 7.20: Precision Graph of English Character Segmentation

The graph illustrates the precision metric for English character segmentation as a function of training epochs. Precision quickly reaches a high level, close to 1.0, within the initial epochs and sustains this peak with very little variance throughout the entire training process up to 50 epochs. This indicates that once the model

learned to correctly segment characters, it continued to do so with high accuracy, consistently identifying true positives while keeping false positives low. The stable line suggests that the model's precision did not fluctuate significantly after the initial training phase, maintaining a high standard of performance.

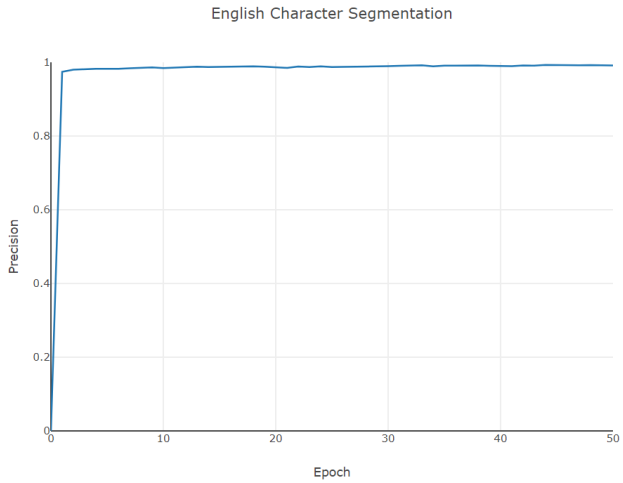


Figure 7.21: Recall Graph of English Character Segmentation

The graph shows the recall metric for English character segmentation, maintaining a high level close to 1.0 throughout the training duration. This indicates that the model is consistently able to identify a high percentage of the relevant characters within the dataset across all epochs. The model's ability to maintain such high recall from the early stages of training suggests that it has a strong capability for segmenting all the relevant characters without missing many actual cases (low false negatives). The stability of the recall metric indicates that the model's sensitivity to detecting characters remained high and unchanged as training progressed.

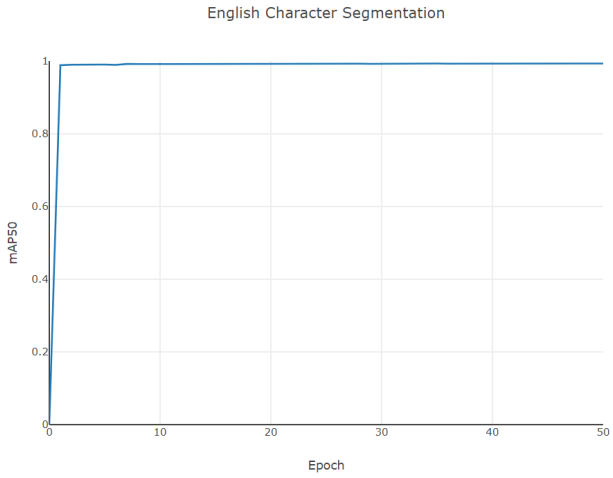


Figure 7.22: mAP50 Graph of English Character Segmentation

The graph depicts a metric performance, specifically mean Average Precision at 50% threshold (mAP50), for the task of English character segmentation over

multiple training epochs. The mAP50 value starts very high, almost at 1.0, from the first epoch, and maintains this level throughout the training process up to epoch 50, as shown by the flat line. This suggests that the model achieved a high level of precision in segmenting English characters from the onset and did not show any significant variation or improvement over time. The consistency in performance could imply that the model was very effective at the segmentation task or that it quickly reached its peak capability early in the training.

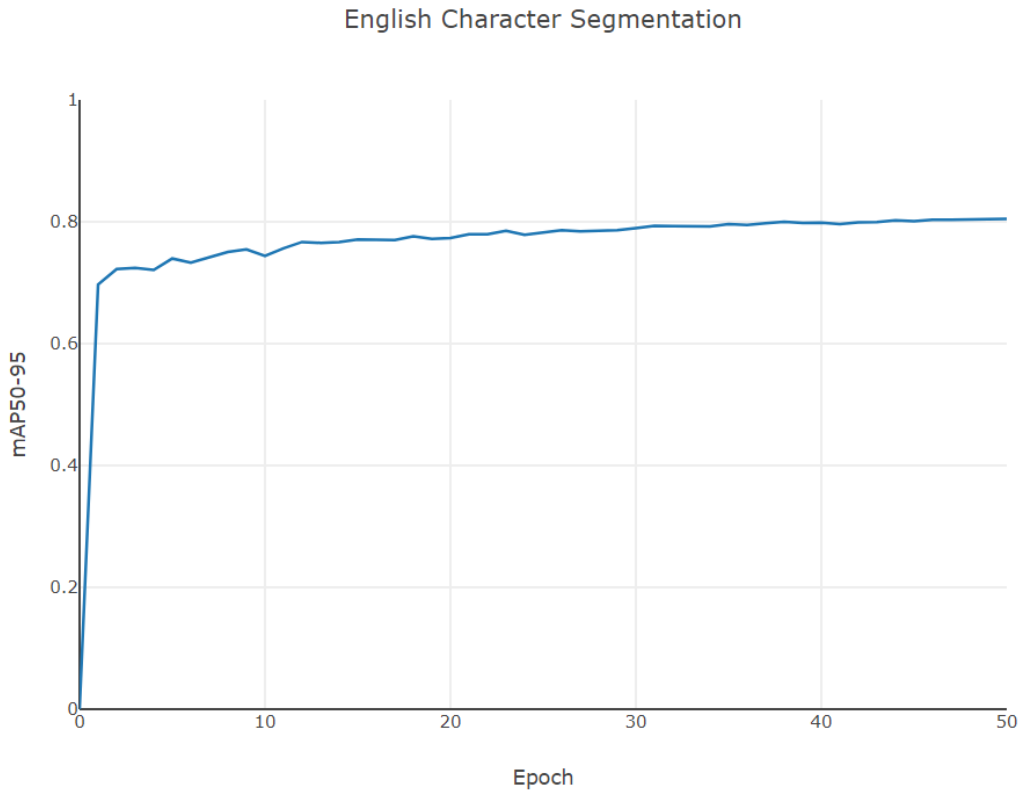


Figure 7.23: mAP50:95 Graph of English Character Segmentation

This graph shows the mean Average Precision at a stricter threshold (mAP50:95) for English character segmentation over a series of training epochs. The mAP50:95 value rises quickly in the initial epochs, indicating rapid learning and significant improvement in the model's ability to segment characters with precision. After this initial rise, the curve flattens, suggesting that the model's performance has plateaued, maintaining a high level of precision from there on. The graph shows minor fluctuations in precision, which is normal in training dynamics, but overall, the performance remains consistently high across the epochs. This indicates that the model has reached a stable level of performance early in the training process.

7.1.1.6 Nepali Character Recognition Model

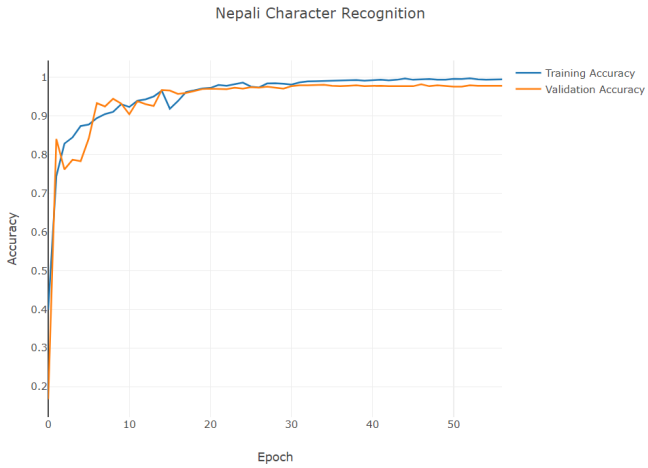


Figure 7.24: Accuracy Plot of Nepali Character Recognition

The graph depicts the accuracy of a model for Nepali character recognition over 50 epochs. Both training and validation accuracies start low but quickly rise, indicating that the model is effectively learning from the data. After the initial steep increase, the accuracies level off, with training accuracy being slightly higher than validation accuracy. This pattern is typical and demonstrates that the model fits well with the training data while still performing adequately on unseen data. The small gap between the two lines suggests that the model is not overfitting significantly. Throughout the training process, the accuracies remain relatively stable, with the validation accuracy showing some variability but still closely tracking the training accuracy, which is a sign of good generalization.

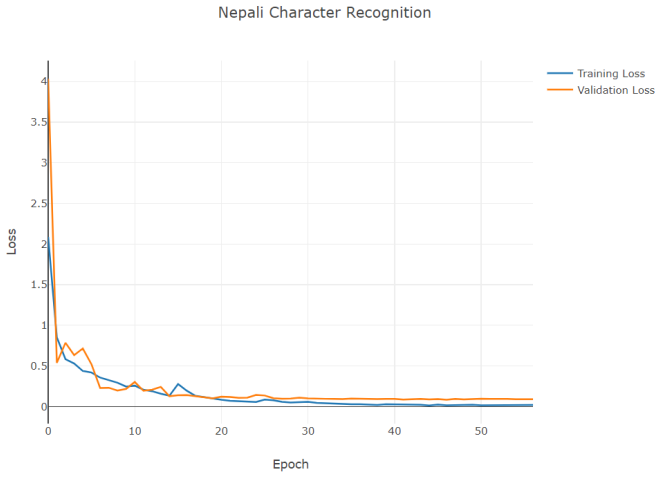


Figure 7.25: Loss Plot of Nepali Character Recognition

The graph shows the loss for a model training on Nepali character recognition, with both training and validation losses starting at high values. These losses sharply decrease in the initial epochs, reflecting rapid initial learning. As the epochs progress, both losses continue to decline at a slower rate and eventually

plateau, indicating that the model is beginning to converge. The training and validation losses closely mirror each other throughout the training process, which suggests that the model is not overfitting and is generalizing well to unseen data. Towards the later epochs, both lines are nearly flat, signifying little to no improvement in loss reduction, which could mean that the model has reached its optimal performance given the current network architecture and training data.

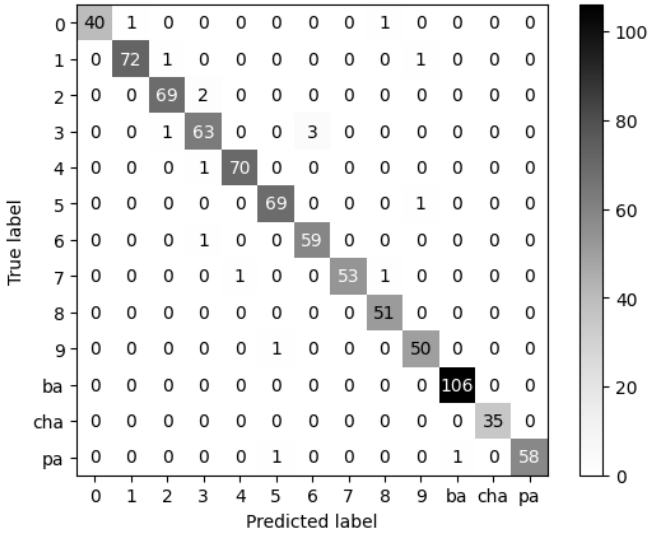


Figure 7.26: Confusion Matrix of Nepali Character Recognition

7.1.1.7 English Character Recognition Model

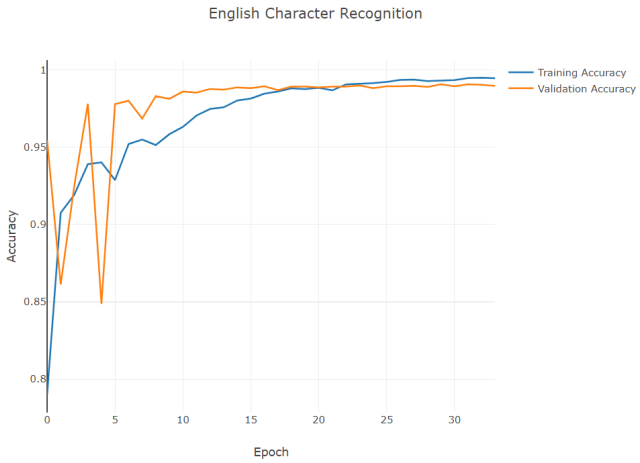


Figure 7.27: Accuracy Plot of English Character Recognition

The accuracy of both training and validation increases and stabilizes after initial fluctuations. The training accuracy is consistently higher than the validation accuracy, which is common as models tend to perform better on the data they have seen. However, the validation accuracy closely follows the training accuracy without a significant gap, indicating that the model is generalizing well and not overfitting. After about 10 epochs, the accuracies of both training and validation

plateau, suggesting that the model has reached its learning capacity with the given data and architecture.

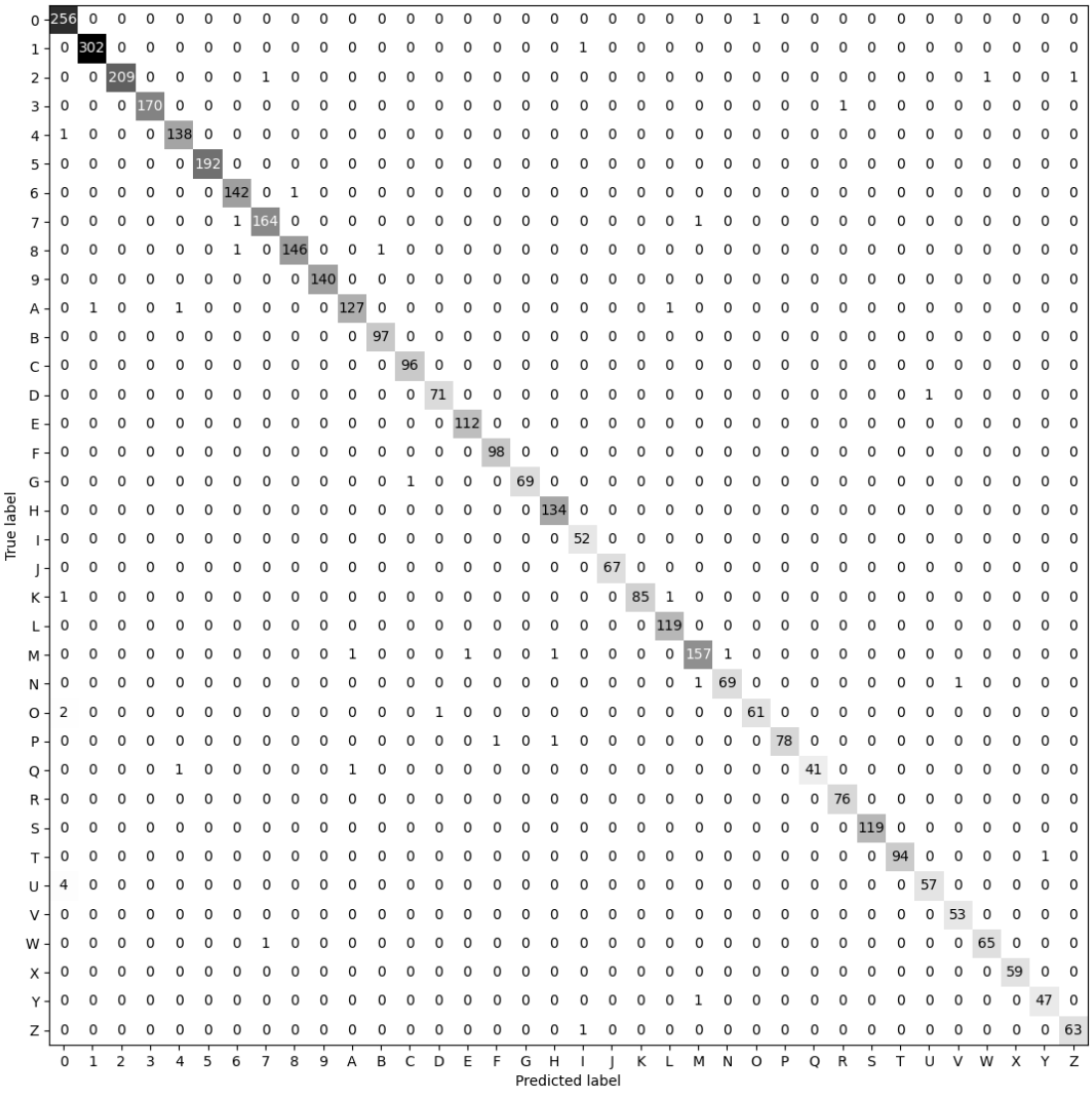


Figure 7.28: Confusion Matrix of English Character Recognition

The graph below shows both training and validation loss decreasing sharply before plateauing as epochs increase, indicating initial rapid learning that stabilizes over time. The convergence of training and validation loss suggests that the model is generalizing well without significant overfitting. After about 10 epochs, both losses exhibit minimal changes, implying additional training epochs might not lead to substantial improvements.

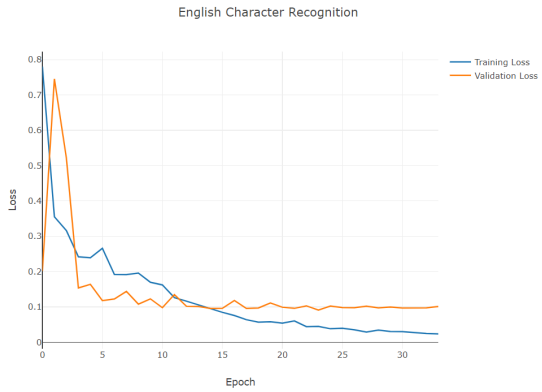


Figure 7.29: Loss Plot of English Character Recognition

7.1.1.8 Summary

Table 7.1: Summary of YOLOv8n Models

Models	**Precision**	**Recall**	**mAP50**
Vehicle Detection	0.9611	0.9878	0.9878
Numberplate Detection	0.9898	0.9800	0.98904
Numberplate Classification	0.9973	0.9710	0.9950
English Segmentation	0.9913	0.9958	0.9935
Nepali Segmenetation	0.9837	0.9766	0.9936

Table 7.2: Summary of Proposed CNN Models

Models	**Training Accuracy**	**Validation Accuracy**
English Character Recognition	0.9878	0.9611
Nepali Character Recognition	0.9898	0.9800

7.1.2 System Testing

After the integration of all the models, the ANPDR system was formed. The system was deployed using AWS. Finally, it needed to be tested with available existing models. The problem with the existing models is that most of these models are unable to work on two types of numberplates i.e. English embossed numberplates and Nepali traditional numberplates. So for the testing, we used the following available models which could detect the given type of numberplates:

Table 7.3: Models for Evaluation

Models	Detect English	Detect Nepali
Tesseract OCR	Yes	Yes
Easy OCR	Yes	Yes
Keras OCR	Yes	No
ANNPR system	No	Yes
ANPR using YOLOv5	Yes	No

For evaluating all of these models with the ANPDR system, we used the designed test case, and the following number of number plates were correctly predicted:

Table 7.4: Test Case Result

Models	Correct English Prediction	Correct Nepali Prediction
Proposed System	96	93
Tesseract OCR	32	5
Easy OCR	21	3
Keras OCR	22	-
ANNPR system	-	59
ANPR using YOLOv5	66	-

We can see that the proposed ANPDR model produced a total accuracy of 94.5% which easily outperforms all the given models in comparison to the test case. The other models that were tested against the system failed to predict the images correctly due to variations in the dataset. We can deduce that the ANPDR system is suitable for the parking system.

7.2 Overall System

The state-of-the-art system incorporates a vehicle detection model, an ANPDR system, and an active mobile application and its databases. The vehicle and numberplate systems are used in correspondence with the mobile application to provide a fully functional Smart Parking Management System for both the user and service operator. This smart parking system works actively for a private parking space under the name Nawadurga Parking. It is also scalable and deployable according to the needs of other operators wanting to use the service app for their own parking space.

7.2.1 Application Walkthrough

This portion of the report is a tour across our application where we navigate through our developed application and discuss its use case.

7.2.1.1 User Panel

- Login Page : Users can log in to their account through this page which is registered under their phone number as observed in 9.2a.
- Overview : Users can view the top view of the parking space to see the overall overview of the parking space as observed in 9.2b.
- Vehicle Logs : Users can view the total hours they parked and also their entry and exit time stamps as observed in 9.3a.
- Payment : Users can view the generated bill and also the payment history. They can also continue with the payment if they would like to do so through an online payment gateway as observed in 9.3b.
- Notification : Users will receive all the important notifications here as observed in 9.4a.
- Profile: A overall profile page from where users can edit their profile and also add or remove vehicles under their profile as observed in 9.4b.
- Edit Profile : Users can edit their profile here as observed in 9.5a.
- Add Vehicle : Users can add their vehicle details here as observed in 9.5b.

7.2.1.2 Admin Panel

- Side Menu : From here admin can access all the menus including an extra alert menu as observed in 9.6b and 9.6a.
- Notification : Admin will receive all the necessary notification here as observed in 9.7a.
- Alert : Admin can view the list of images of the unauthorized vehicle that tried to enter the parking space along with the time and date stamp of the entry as observed in 9.7b.
- Alert Image : Admin can open the image in alert section in a full screen mode as observed in 9.8a.
- Profile Management : Admin can add a user and manage them. There is also a Search menu for convenience as observed in 9.8b.
- Add User : Page to add a user as observed in 9.9a.
- View User : Page for viewing a user as observed in 9.9b.
- User Configuration : Admin can manage a user here and also view the vehicle logs as observed in 9.10a.
- User Vehicle Configuration : Admin can allocate a user vehicle to a slot number as observed in 9.10b.
- Search User : Admin can search user based on various parameters for convenience as observed in 9.11a.

Chapter 8

Conclusion

This project was undertaken to construct a Smart Parking Management System using various YOLOv8 and CNN models which can be used in real-world applications. The system can be used seamlessly in the parking system to detect parking occupancy and recognize both types of existing numberplates in Nepal with an accuracy of 94.5%. The mobile application-based system that comprises all the integrated models was developed as the final output of the project.

Chapter 9

Limitations and Future Enhancements

9.1 Limitations

Despite our best efforts in the development of the project, the project has the following limitations:

- The system can only work with private Nepali traditional numberplates.
- The system cannot work on two-wheeler vehicle occupancy.
- The proposed system requires an on-site server.
- The numberplate detection is unable to work with the night vision camera feed.

9.2 Future Enhancements

As for future enhancements, the following can be considered:

- Expand the working of the model with the public as well as government vehicle number plates.
- Expand the vehicle occupancy system to two-wheeler vehicles as well.
- Diversify the number plate detection system into other application areas such as over-speed detection, and traffic surveillance.
- Introduce dynamic pricing strategies based on parking demand and availability.

Bibliography

[1] "Introduction to yolo algorithm for object detection," accessed: May 15, 2024. [Online]. Available: https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/grids.png

[2] "Intersection over union (iou) equation," accessed: May 15, 2024. [Online]. Available: https://b2633864.smushcdn.com/2633864/wp-content/uploads/2016/09/iou_equation.png?lossy=2&strip=1&webp=1

[3] [Online]. Available: https://b2633864.smushcdn.com/2633864/wp-content/uploads/2016/09/iou_examples.png?lossy=2&strip=1&webp=1

[4] [Online]. Available: https://velog.velcdn.com/images/peterkim/post/516e83d2-c712-4fed-b922-0f8c01b9c138/image.png

[5] [Online]. Available: https://www.proprofsproject.com/blog/project-life-cycle-and-its-phases

[6] R. K. Gupta and G. Rani, "Machine learning and iot based real time parking system: Challenges and implementation," 04 2020.

[7] P. Melnyk, S. Djahel, and F. Naït-Abdesselam, "Towards a smart parking management system for smart cities," 09 2019.

[8] A. Khanna and R. Anand, "Iot based smart parking system," 01 2016.

[9] T. Lotlikar, M. Chandrahasan, A. Mahadik, M. Oke, and A. Yeole, "Smart parking application," *International Journal of Computer Applications*, vol. 149, pp. 32–37, 09 2016.

[10] R. Widyasari, M. Candra, and S. Akbar, "Iot-based smart parking system development," 11 2019, pp. 1–6.

[11] M. Fraifer and M. Fernström, "Investigation of smart parking systems and their technologies," 12 2016.

[12] S. Lee, D. Yoon, and A. Ghosh, "Intelligent parking lot application using wireless sensor networks," in *2008 International Symposium on Collaborative Technologies and Systems*, 2008, pp. 48–57.

[13] H.-N. Manh, "Rethinking feature generalization in vacant space detection," *Sensors*, vol. 23, p. 4776, 05 2023.

[14] X. Ding and R. Yang, "Vehicle and parking space detection based on improved yolo network model," *Journal of Physics: Conference Series*, vol. 1325, p. 012084, 10 2019.

[15] E. Prem, C. Roy, A. Thapa, K. Shrestha, P. Karmacharya, and R. Karna, "Vehicle number plate recognition and parking system," 12 2018.

[16] M. G. P. D. G. V. S. G. CH. Ranga Reddy, Angshuman Roy, "Vehicle detection using yolov3 for counting the vehicles and traffic analysis."

[17] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning," in *2016 IEEE Symposium on Computers and Communication (ISCC)*, 2016, pp. 1212–1217.

[18] S. Amarasooriya, P. Peiris, and D. Herath, "Implementation of smart parking system using image processing," 03 2023.

[19] M. Rashid, A. Musa, A. Rahman, M. S. Nur Farahana, and A. Farhana, "Automatic parking management system and parking fee collection based on number plate recognition," *M. M. Rashid, A. Musa, M. Ataur Rahman, and N. Farahana, A. Farhana*, vol. 2, pp. 93–98, 01 2012.

[20] K. Kumar, V. Singh, L. Raja, and S. Bhagirath, "A review of parking slot types and their detection techniques for smart cities," *Smart Cities*, vol. 6, pp. 2639–2660, 10 2023.

[21] A. Raj, V. S. Selvaraj, M. Priya, D. Selvaraj, P. B, and M. Srinivasan, "Intellipark -smart parking system," 04 2023.

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 06 2015.

[23] "Indian vehicle numberplate yolo dataset." [Online]. Available: https://www.kaggle.com/datasets/deepakat002/indian-vehicle-number-plate-yolo-annotation

[24] "German license plate dataset." [Online]. Available: https://universe.roboflow.com/max-mustermann-gmm7j/german-license-plates-hptbz

[25] "Nepali numberplate dataset." [Online]. Available: https://universe.roboflow.com/anish-shilpakar/ocr-6vmey

Appendix

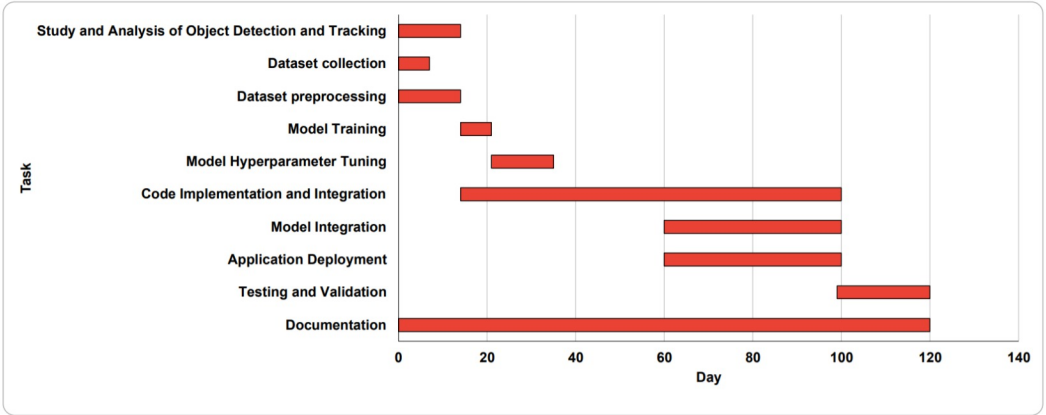
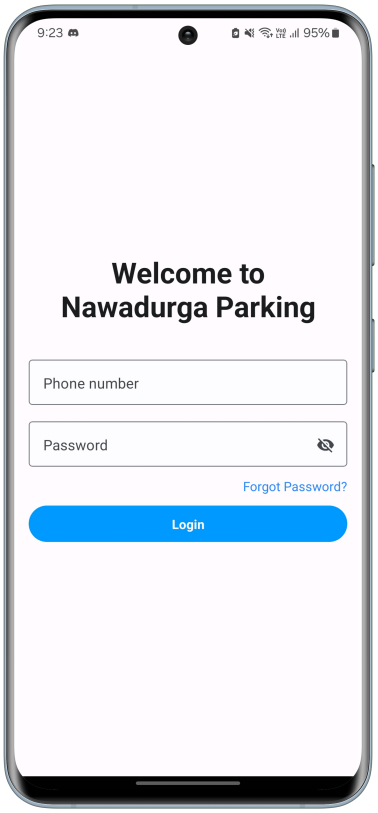
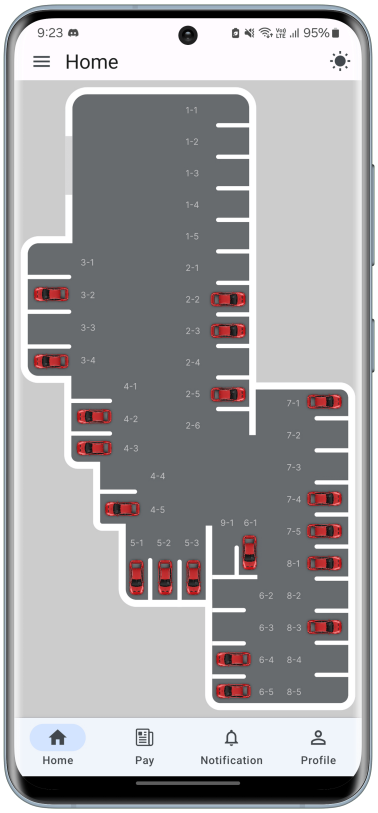


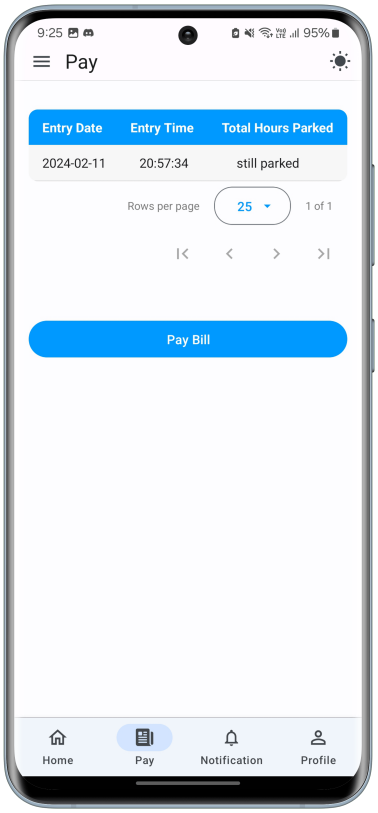
Figure 9.1: Gantt-Chart



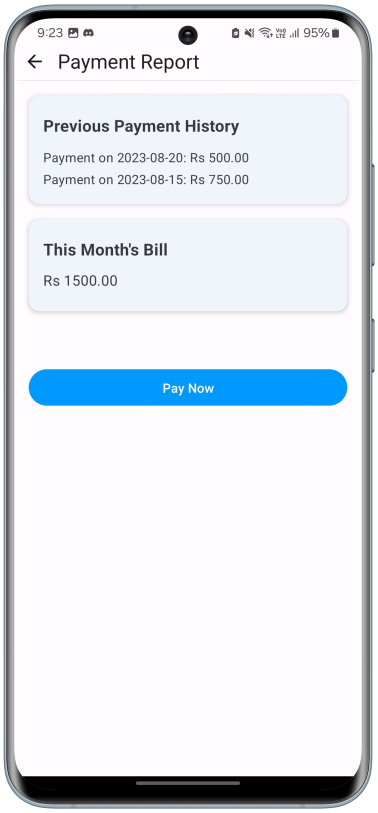
(a) Login Page



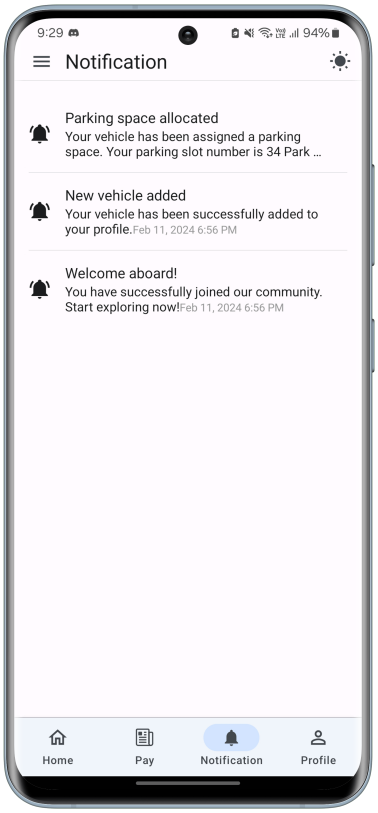
(b) Overview



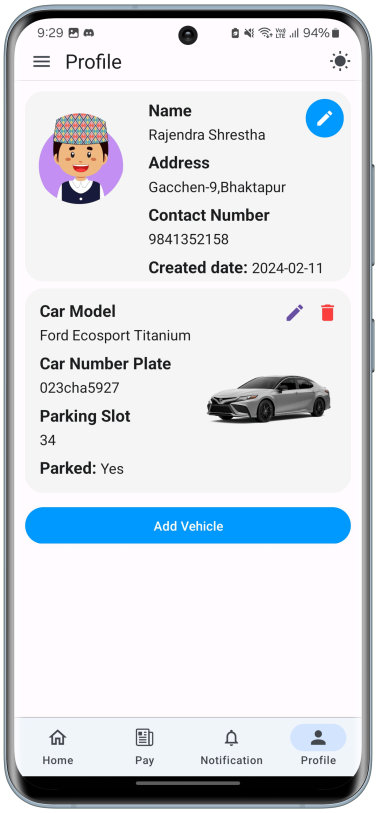
(a) Payment/Vehicle log



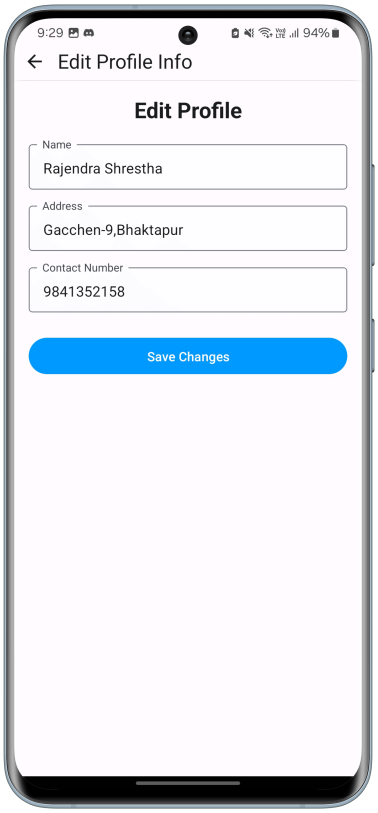
(b) Payment Page



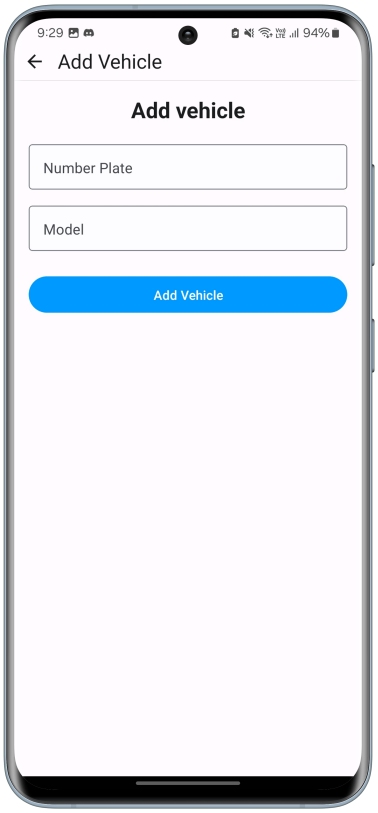
(a) Notification



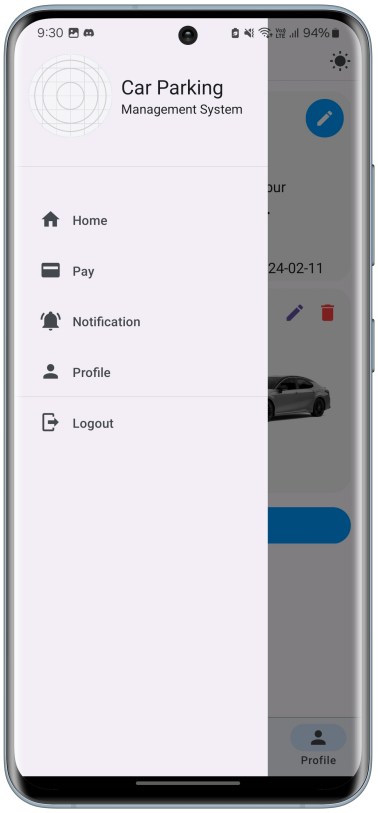
(b) Profile



(a) Edit Profile

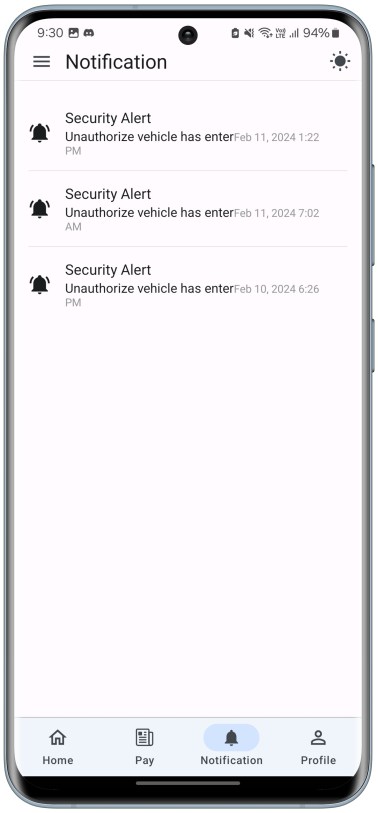


(b) Add vehicle



(a) Side Menu (Light Mode)

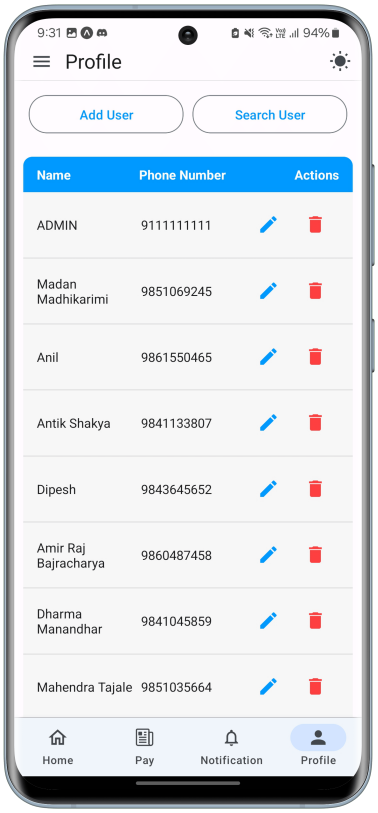
(b) Side Menu (Dark Mode)



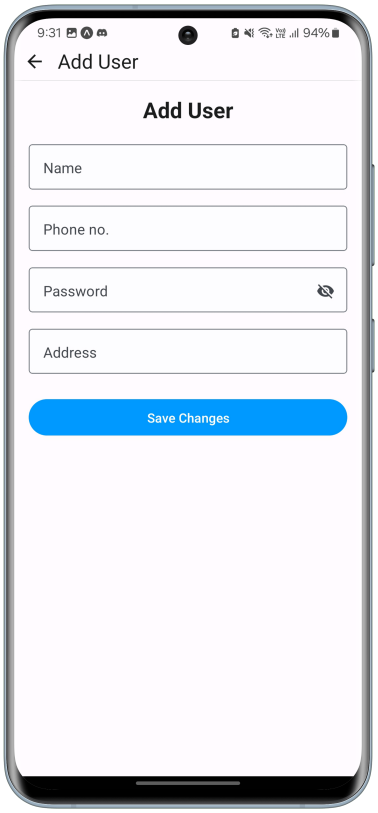
(a) Notification

(b) Alert

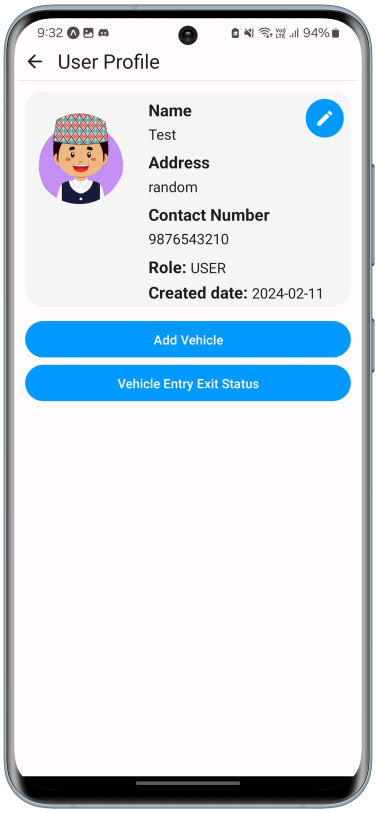
(a) Alert Image



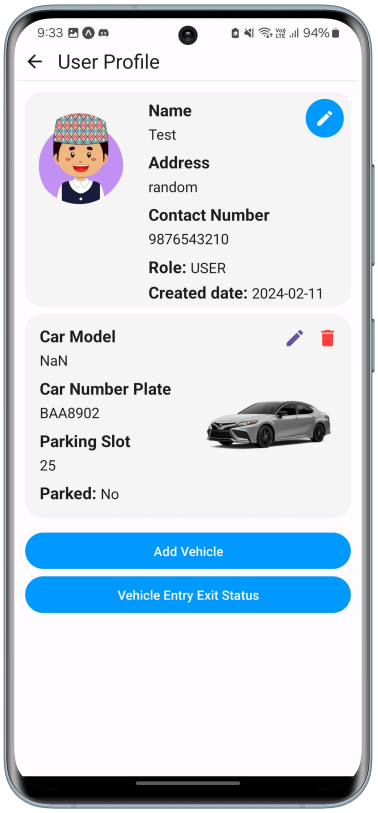
(b) Profile Management



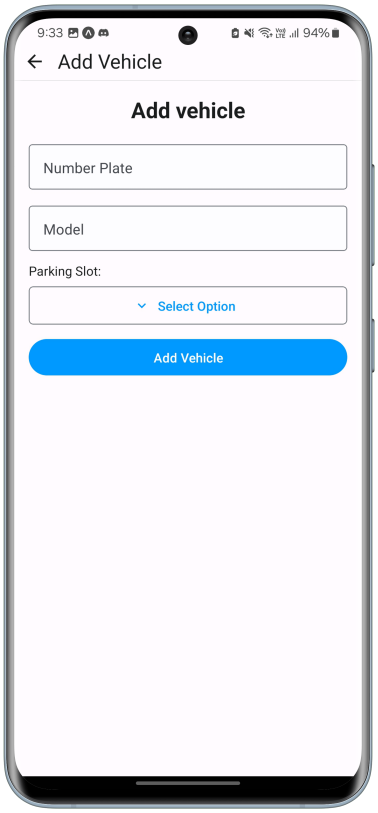
(a) Add User



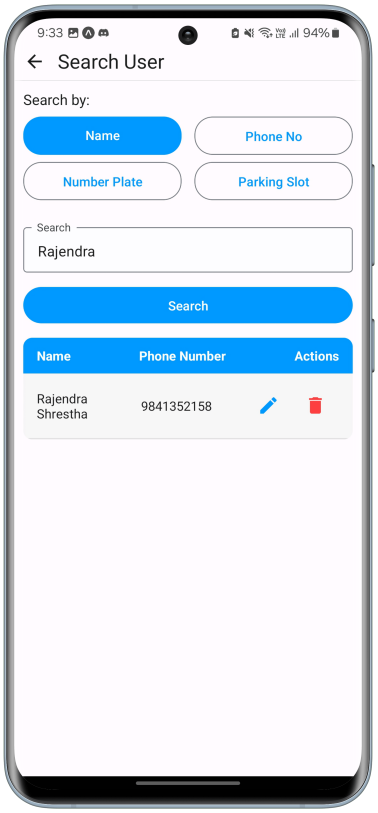
(b) View User



(a) User Configuration



(b) User Vehicle Configuration



(a) Search User