“PARKEASE - PARKING SLOT DECTECTION AND MANAGEMENT”

A PROJECT WORK- II REPORT SUBMITTED TO

THE NATIONAL INSTITUTE OF ENGINEERING

(An Autonomous Institution Under VTU)

In partial fulfillment of the requirements for project work- II, eighth semester

Bachelor of Engineering

In

Computer Science & Engineering

Submitted By

Aditya Kiran(4NI21CS005) **Ajay S Biradar(4NI21CS009)**

Arjun Nambiar(4NI21CS022) **Ashish Baghel(4NI21CS024)**

Under The Guidance Of

Mrs. Poornima N

Assistant Professor (FTC)

Department of CS&E, NIE

Mysore-570008

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE NATIONAL INSTITUTE OF ENGINEERING

(An Autonomous Institution Under VTU)

Mananthavadi Road, Mysuru- 570008

THE NATIONAL INSTITUTE OF ENGINEERING

(An Autonomous institution, affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Mysore -570008

CERTIFICATE

This is to Certify that the project work entitled "PARKEASE - parking slot dectection and management**"** is a bonafide work carried out by **Aditya Kiran** (4NI21CS005), **Ajay S Biradar** (4NI21CS009), **Arjun Nambiar** (4NI21CS022) and **Ashish Baghel** (4NI21CS024) in fulfillment for project work–II, eighth semester, Computer Science and Engineering**,** The National Institute of Engineering (Autonomous under VTU) during the academic year 2024–2025.It is certified that all corrections and suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the department library. The project work – II report has been approved in fulfillment as per academic regulations of The National Institute of Engineering, Mysuru**.**

Signature of the Guide	**Signature of Co-Guide**	**Signature of the H.O.D**
(Mrs. Poornima N)	**(Mrs. Usha K Patil)**	**(Dr. Anitha R)**
Assistant Professor (FTC),	**Assistant Professor**	**Professor and Head,**
Dept. of CS&E,	**Dept. of CS&E,**	**Dept. of CS&E,**
NIE, Mysuru	**NIE, Mysuru**	**NIE, Mysuru**

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible whose constant guidance and encouragement crown all efforts.

First and foremost, we would like to thank our beloved principal **Dr. Rohini Nagapadma** for being the patron and the beacon light for this project.

We would like to express our sincere gratitude to our HOD **Dr. Anitha R,** Professor and Head, Department of CSE, NIE for her relentless support and encouragement.

It gives us immense pleasure to thank our guide **Mrs. Poornima N,** Assistant Professor, Department of CSE, NIE for her valuable suggestions and guidance during the process of the project and for having permitted us to pursue work on the subject.

-Aditya Kiran
-Ajay S Biradar
-Arjun Nambiar
-Ashish Baghel

TABLE OF CONTENT

LIST OF FIGURE

Chapter 1

INTRODUCTION

In the bustling, fast-paced environment of modern cities, the challenge of finding parking spaces has become a significant issue, contributing to traffic congestion, wasted fuel, and increased stress levels for drivers. ParkEase is designed to address these problems by offering an innovative solution that leverages cutting-edge technologies to optimize urban parking management. This system integrates computer vision with real-time data, allowing users to easily locate available parking spaces and enabling parking managers to efficiently control and monitor parking areas.

ParkEase combines the power of OpenCV for detecting parking slots with a mobile application built on React Native, offering seamless, real-time updates to users. With a tech stack that includes Python, TypeScript, JavaScript, the system ensures high-performance data processing and communication between the frontend and backend. The use of MongoDB for scalable data storage, together with Prisma as an ORM, ensures smooth and efficient database management.

Designed to be versatile and scalable, ParkEase supports multiple operating systems—Windows, Linux, iOS, Android, and Raspbian-making it adaptable to various environments. The system is equipped to run on standard hardware, with the inclusion of a Raspberry Pi and a camera module for real-time parking slot detection and tracking. With its intelligent, data-driven approach, ParkEase aims to transform urban parking into a streamlined, efficient, and stress-free experience for both parking managers and users alike.

Chapter 2

SYSTEM ANALYSIS

2.1 LITERATURE SURVEY

2.1.1 Research Papers on Existing System

[1] Smart Parking Systems Using IoT and Machine Learning for Parking Slot Detection, S. Jain, S. Bhardwaj, and M. Pandey, International Conference on Computational Science and Engineering (CSE), 2021.

Description: This paper presents an IoT-based smart parking system integrated with machine learning for detecting and managing parking slots. The proposed system focuses on optimizing the use of parking spaces in urban areas by using sensors to detect the occupancy status of parking spots. The data is processed in real-time and provides users with information about available parking slots via a mobile application. The system also aims to reduce parking search time and traffic congestion. However, the integration of computer vision for slot detection is limited, as the system relies primarily on sensor data.

[2]Automated Parking Slot Management System Using Machine Learning and Big Data, A. Das, R. Kumar, and P. Khatri, IEEE International Conference on Computing, Communication, and Automation (ICCCA), 2019.

Description: This paper proposes an automated parking slot management system that utilizes machine learning for real-time parking slot allocation and monitoring. The system integrates big data analytics to process and predict parking demand patterns, improving parking management in congested areas. It focuses on integrating data from various parking lots and providing users with optimal parking suggestions. The study highlights the challenges of managing large datasets and the need for robust database management solutions, but it does not explore the use of computer vision for slot detection.

[3]Parking Slot Detection Using Computer Vision and Deep Learning, M. Ibrahim, A. Hosny, and S. Abdul, IEEE International Conference on Advanced Robotics and Intelligent Systems, 2020.

Description : This research introduces a parking slot detection system that leverages computer vision techniques combined with deep learning models. By utilizing image data from surveillance cameras, the system can identify vacant and occupied parking spaces in real-time. OpenCV and TensorFlow are used to implement the detection models, which have demonstrated high accuracy in controlled environments. The paper highlights the importance of real-time updates to ensure seamless communication with mobile apps for user notifications but lacks a focus on backend management or system scalability.

2.1.2. EXISTING SYSTEM AND ITS DRAWBACKS:

Existing parking management systems typically rely on various methods such as manual spot management, sensors, or RFID-based solutions to monitor and allocate parking spaces. While these systems have seen widespread use, they come with notable limitations. Manual systems are prone to human error, leading to inaccurate data and inefficient space allocation. Sensor-based systems, though more automated, face scalability challenges and are often costly to implement and maintain across large urban areas. Additionally, they can be less reliable due to sensor malfunctions, environmental conditions, or misdetection, causing further operational inefficiencies.

Moreover, many existing systems fail to offer real-time updates to users, resulting in delays or misinformation about available parking spaces. This can lead to a frustrating user experience, increased search time, and worsened traffic congestion. These systems also tend to have limited integration capabilities with mobile applications and are not optimized for cross-platform compatibility, hindering their overall utility in modern, tech-driven urban environments.

Overall, the drawbacks of these systems include high costs, limited real-time communication, scalability issues, and a lack of advanced features such as computer vision for more accurate parking slot detection. This makes them inefficient for growing urban centers where the demand for optimized parking solutions is increasing.

2.1.3. PROPOSED SYSTEM:

The proposed system, ParkEase, addresses the limitations of existing parking management solutions by integrating computer vision, machine learning, and real-time data processing. Leveraging OpenCV for parking slot detection, this system offers a more scalable and accurate solution for tracking parking space availability. Unlike sensor-based systems, which may fail in harsh environmental conditions, the use of cameras combined with computer vision ensures reliable detection across different scenarios.

ParkEase also incorporates a dynamic mobile application, built using React Native, that provides users with real-time updates on available parking slots. Through this app, users can view, reserve, and navigate to parking spaces with ease. The backend is supported by a GraphQL API, ensuring efficient data querying and seamless communication between the frontend and backend components. MongoDB and Prisma are used for scalable database management, allowing the system to handle a large volume of data from multiple parking locations.

This system also features a dedicated interface for parking managers to monitor parking activity and dynamically create or modify parking slots. By integrating machine learning techniques, the system can predict peak parking times and optimize space usage, further improving parking efficiency.

Overall, the proposed system aims to reduce search times, optimize space usage, and enhance user satisfaction through a combination of real-time updates, cross-platform compatibility, and intelligent management tools.

2.2 SYSTEM REQUIREMENTS

2.2.1 Hardware Requirement

Operating System : Windows/Linux

Processor : Dual core, 64-bit processor

Memory : 4GB or higher

Additional Devices : Raspberry Pi, Camera Module rev 1.3

2.2.2 Software Requirement

Programming Language : Python, TypeScript, JavaScript

Tools : Jupyter Notebook, OpenCV, Prisma , MongoDB, sklearn, React Native, numpy

Chapter 3

SYSTEM DESIGN

3.1 System Architecture

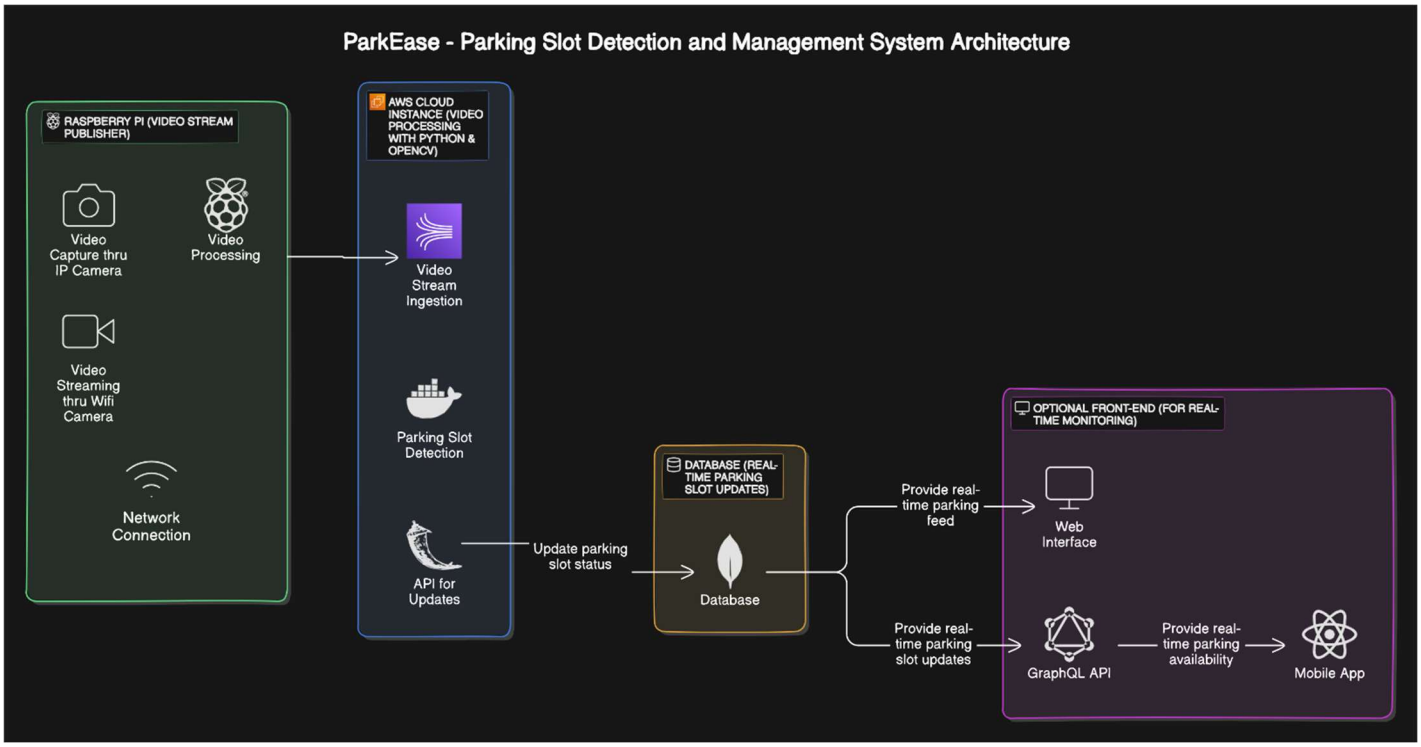

Fig 1: System architecture

3.2 Database Schema/ ER Diagram

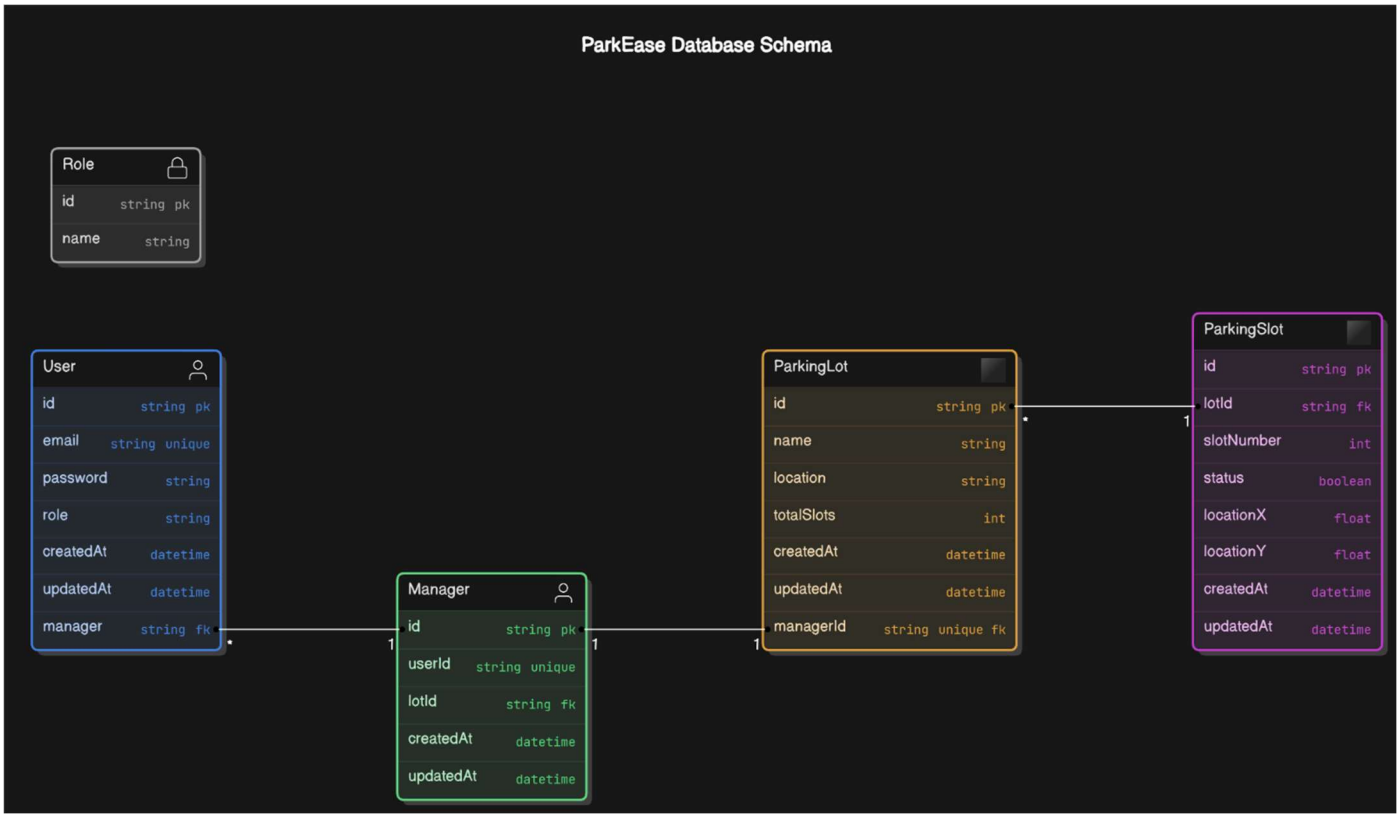

Fig 2: Database Schema/ER Diagram

3.3 Architecture Overview:

Tech Stack:

- **Backend**: Flask, Prisma, OpenCV, Numpy, PyTorch, MongoDB
- **Frontend**: Tailwind CSS, ShadCN UI
- **Cloud Services**: AWS Kinesis, AWS EC2
- **Device**: Raspberry Pi running Raspbian
- **Video Streaming**: OpenCV, AWS Kinesis
- **Others**: AWS EC2 for processing, MongoDB for storage

1. Raspberry Pi (Video Stream Publisher)

Function: Captures video from a camera (e.g., PiCamera or webcam) and streams it to the cloud for further processing.

Role: Acts as the client, responsible for capturing real-time video feed and transmitting it to the cloud server, where computationally expensive operations (like parking slot detection) are carried out.

Key Components:

- **Video Capture**: Raspberry Pi uses OpenCV or PiCamera to capture live video.
- **Streaming**: The video stream is sent using GStreamer, FFmpeg, or MJPEG Streamer. Streaming formats include RTSP, HLS, or WebRTC for low-latency transmission.
- **Communication**: Communication with the cloud is handled through protocols like HTTP, WebSocket, or MQTT for lightweight IoT communication. The Pi pushes the video feed to an Amazon EC2 instance (or any cloud server).

2. Cloud Instance (Video Processing Unit)

Function: Processes the video feed, detects parking slot occupancy, and updates the status in real time.

Role: This unit performs heavy computations like video processing and parking slot detection using Python and OpenCV. The Raspberry Pi sends the video feed to the cloud instance, which processes it and updates the parking availability status.

Key Components:

- **Video Stream Ingestion**: Video streams are received on the cloud instance using FFmpeg, GStreamer, or AWS Kinesis Video Streams.
- **Parking Detection Algorithm**: Python code using OpenCV and Numpy processes the video feed to recognize parking slots and detect occupancy in real time. A neural network, trained with PyTorch, could enhance the accuracy of detecting parked cars.
- **API for Status Updates**: A Flask or FastAPI API sends parking slot status updates to a backend database, updating availability when a parking slot is filled or emptied.
- **Optional GPU Usage**: For enhanced performance, an Amazon EC2 instance with NVIDIA GPUs can be employed to speed up video processing using CUDA libraries.

3. Parking Slot Detection and Status Update

Function: Detects whether parking slots are free or occupied based on real-time video processing and updates the system accordingly.

Role: The detection algorithm analyzes the video feed and determines whether specific parking slots are occupied. If there is a status change (from free to occupied or vice versa), the system triggers an API call to update the database.

Key Components:

- **Detection Algorithm**: OpenCV-based Python algorithm to identify and track parking slots and car positions.
- **Status Update API**: Upon detecting a change in parking slot availability, the system triggers an HTTP or GraphQL API request to update the parking slot status in the cloud database.

4. Database (Real-Time Parking Slot Updates)

Function: Store and manage real-time parking slot availability, providing data for real-time monitoring.

Role: The database keeps a real-time record of the parking slot status (occupied/free) and serves this data to the front-end interface.

Key Components:

- **Database**: Use MongoDB to store parking slot information and provide real-time access to the front-end.

- **Updates**: The cloud instance communicates directly with the database to update the status of parking slots. For event-driven architectures, AWS Lambda or AWS SQS can be used to manage updates asynchronously.

5. Communication Workflow

1. **Raspberry Pi (Video Capture)**: Captures live video from a camera and streams it to the cloud server using RTSP, HTTP, or WebRTC.
2. **Cloud Instance (Video Processing)**: Receives the video stream, processes it using OpenCV, and detects parking slot status (occupied or free).
3. **Cloud Instance (API Call)**: Upon detecting a status change, the instance sends a request to update the parking slot status in the backend database.
4. **Database**: The MongoDB database stores and manages parking slot availability in real time.

6. Optional Front-End (Real-Time Monitoring)

Function: Display parking slot availability on a web or mobile interface in real time.

Role: Provides a user-friendly interface for monitoring parking availability, which could be used by administrators or end users.

Key Components:

- **Front-End**: Use React.js and Tailwind CSS for building a clean, responsive UI. ShadCN UI can enhance the UI/UX with ready-to-use components.
- **Real-Time Updates**: Use WebSockets or Server-Sent Events (SSE) to push parking slot updates to the front-end in real time, ensuring live monitoring for users.