KONGU ENGINEERING COLLEGE

(Autonomous)

Perundurai, Erode – 638 060

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND

DATA SCIENCE

VEHICLE PARKING SYSTEM

AN APPLICATION PROJECT REPORT

FOR WEB TECHNOLOGY

Submitted by

S. SATHIYASEELAN (22ADR096)

D.PAVUL CHINNAPPAN (22ADR076)

S. SRINESH (22ADR103)

TABLE OF CONTENT

1.ABSTRACT

2.INTRODUCTION

2.1. Need for a Comprehensive Solution
2.2. Proposed Solution Overview
2.3. Objectives

3.TECHNOLOGY USED

3.1. Front End
3.2. Back End

4.IMPLEMENTATION

4.1. Initial Setup
4.2. Check-In and Slot Allocation
4.3. Check-Out and Payment
4.4. Fetch Slot of Vehicle
4.5. Slot Status Monitoring

5.CODE

5.1 Frontend
5.2 Backend
5.2.1 Module
5.2.2 Controller
5.2.3 Route

6.RESULTS

7.CONCLUSION

1.ABSTRACT

The Vehicle Parking System is designed to enhance parking management using modern technology. It provides an easy-to-use platform for users to quickly find and reserve available parking spots. During check-in, the system allocates an available slot and records the vehicle's details and in-time, ensuring efficient tracking. At check-out, the system calculates the parking fee based on the duration of stay, generating a detailed report that includes in-time, out-time, and the total amount due. This ensures a transparent and straightforward payment process.

Additionally, the system offers real-time monitoring of parking slots, showing which spots are occupied and which are available. This feature helps users find parking easily and allows administrators to manage parking spaces more effectively. The system also includes a slot status feature, which provides a comprehensive view of how many slots are filled and how many are available at any given time.

By simplifying the check-in, slot allocation, and payment processes, the Vehicle Parking System aims to improve the overall parking experience for users. Future enhancements could include predictive analytics to forecast parking demand and integration with smart city systems to further optimize parking management. This system represents a significant step towards more organized and user-friendly parking solutions.

2.INTRODUCTION

The Vehicle Parking System is designed to streamline and simplify parking management through the use of modern technology. This system provides users with an efficient way to find, reserve, and pay for parking spots, enhancing overall user experience and operational efficiency.

2.1 Need for a Comprehensive Solution

Traditional parking systems are often inefficient and time-consuming, leading to user frustration and mismanagement of parking spaces. A comprehensive solution is needed to automate and optimize the process, providing real-time updates and efficient slot allocation

Add Vehicle in Slot: When a vehicle enters the parking area, the system assigns an available slot and records the vehicle's details along with the check-in time. This automated process ensures accurate tracking and efficient use of parking space.

Remove Vehicle from Slot: Upon exiting, the system calculates the parking fee based on the check-in and check-out times. It generates a detailed report, ensuring a transparent and straightforward payment process, and frees up the slot for the next user.

Fetch Slot of Vehicle: Users can easily find where their vehicle is parked through the system. By entering the vehicle details, the system provides the exact location of the parked vehicle, saving time and effort.

Check Parking Status: The system offers real-time updates on the status of parking slots, indicating which spots are occupied and which are available. This feature helps users quickly find available parking and allows administrators to manage the parking area more effectively.

2.2. Proposed Solution Overview

The Vehicle Parking System leverages modern technology, with a React frontend and an Express.js and MongoDB backend, to streamline parking management. It assigns slots during check-in, calculates fees based on in-time and out-time at check-out, and provides real-time slot status updates. Users can easily find their vehicles and check slot availability, enhancing convenience and efficiency. Future improvements may include predictive analytics and smart city integration for further optimization.

2.3. Objectives

- Streamline the allocation and utilization of parking spaces to minimize congestion and reduce the time spent searching for available spots.
- Provide a user-friendly interface for easy check-in, check-out, and real-time updates on parking slot availability, ensuring a smooth and hassle-free experience.
- Automate the calculation of parking fees based on in-time and out-time, offering transparent and precise billing for users.
- Enable administrators to monitor slot status and manage parking resources effectively, improving overall operational efficiency.

3.TECHNOLOGY USED

3.1. Front End:

- ☐ **React + Vite**: Utilized for frontend development, React offers a fast and responsive user interface, while Vite enhances development speed with quick build times and modern JavaScript module handling.

- ☐ **Tailwind CSS**: Chosen for styling, Tailwind CSS provides utility-first classes for rapid UI development, enabling quick prototyping and consistent design across the application.

3.2. Back End:

- **MongoDB:** Employed as the backend database, MongoDB offers flexibility with its NoSQL document-oriented structure, allowing efficient storage and retrieval of restaurant data.

- **Express.js**: Used for backend development, Express.js provides a minimalist framework for building robust APIs and handling HTTP requests, ensuring seamless communication between the frontend and MongoDB backend.

- **CORS**: The backend uses CORS (Cross-Origin Resource Sharing) to allow the frontend and backend to communicate securely across different domains. This ensures that the frontend can make API requests to the backend without any cross-origin restrictions, facilitating seamless data exchange.

4.IMPLEMENTATION

4.1 Initial Setup:

The system starts by setting up the backend with Express.js and MongoDB for data storage. Necessary dependencies are installed, and the project structure is organized to manage the parking data efficiently.

4.2 Check-In and Slot Allocation:

During check-in, the system assigns an available parking slot to the vehicle and records the vehicle's details along with the check-in time. This ensures that each vehicle is tracked accurately and slots are used efficiently.

4.3 Check-Out and Payment:

At check-out, the system calculates the parking fee based on the in-time and out-time. It generates a detailed report showing the duration of the stay and the total amount due, simplifying the payment process for users.

4.4 Fetch Slot of Vehicle:

Users can find their parked vehicles by entering the vehicle details into the system. The system then provides the exact location of the vehicle, making it easy for users to locate their cars.

4.5 Slot Status Monitoring:

The system provides real-time updates on the status of parking slots, indicating which spots are filled and which are available. This helps users find available parking quickly and allows administrators to manage the parking area effectively.

5.CODE

5.1 FRONTEND

HOME.JSX:

```
import { useState } from 'react';
import CheckIn from './CheckIn';
import CheckOut from './CheckOut';
import SlotStatus from './SlotStatus';
import FindVehicleSlot from './FindVehicleSlot';

const Home = () => {
    const [view, setView] = useState('home');

    return (
        <div className="min-h-screen bg-gray-100 flex flex-col items-center
justify-center p-4">
            <h1 className="text-4xl font-bold text-center mb-8 text-gray-800">Vehicle
Parking Management</h1>
            {view === 'home' && (
                <div className="space-x-4 flex flex-wrap justify-center">
                    <button
                        onClick={() => setView('checkin')}
                        className="m-2 px-6 py-3 bg-blue-500 text-white rounded-full
shadow-lg hover:bg-blue-700 hover:shadow-xl focus:outline-none focus:ring-2
focus:ring-blue-400 transition"
                    >
                        Check In
                    </button>
                    <button
                        onClick={() => setView('checkout')}
                        className="m-2 px-6 py-3 bg-green-500 text-white rounded-full
shadow-lg hover:bg-green-700 hover:shadow-xl focus:outline-none focus:ring-2
focus:ring-green-400 transition"
                    >
                        Check Out
                    </button>
                    <button
                        onClick={() => setView('status')}
                        className="m-2 px-6 py-3 bg-yellow-500 text-white rounded-full
shadow-lg hover:bg-yellow-700 hover:shadow-xl focus:outline-none focus:ring-2
focus:ring-yellow-400 transition"
                    >
                        Check Slot Status
                    </button>
                    <button
                        onClick={() => setView('find')}
                        className="m-2 px-6 py-3 bg-red-500 text-white rounded-full
shadow-lg hover:bg-red-700 hover:shadow-xl focus:outline-none focus:ring-2
focus:ring-red-400 transition"
                    >
                        Find Vehicle Slot
                    </button>
                </div>
            )}
            {view === 'checkin' && <CheckIn />}
            {view === 'checkout' && <CheckOut />}
            {view === 'status' && <SlotStatus />}
            {view === 'find' && <FindVehicleSlot />}
            {view !== 'home' && (
                <button
                    onClick={() => setView('home')}
                    className="mt-4 px-6 py-3 bg-gray-500 text-white rounded-full
shadow-lg hover:bg-gray-700 hover:shadow-xl focus:outline-none focus:ring-2
focus:ring-gray-400 transition"
                >
                    Back
                </button>
            )}
        </div>
    );
};

export default Home;
```

CHECKIN.JSX:

```
import { useState } from 'react';
import axios from 'axios';
import { jsPDF } from 'jspdf';

const CheckIn = () => {
    const [vehicleNumber, setVehicleNumber] = useState('');
    const [message, setMessage] = useState('');
    const [vehicleData, setVehicleData] = useState(null);

    const handleSubmit = async (e) => {
        e.preventDefault();
        try {
            const response = await axios.post('http://localhost:5000/api/vehicles', {
vehicleNumber });
            setMessage(
        <div
         className="p-4 mb-4 text-sm text-green-700 bg-green-100 rounded-lg
dark:bg-green-200 dark:text-green-800"
         role="alert"
        >
         <span className="font-medium">Success!</span> Vehicle checked in
successfully! Slot number: {response.data.slot}.
        </div>
       );
      setVehicleData(response.data);
    } catch (error) {
      setMessage('Error checking in vehicle');
    }
  };

  const generateInvoice = () => {
    if (!vehicleData) return;

    const doc = new jsPDF();
    const date = new Date(vehicleData.entryTime).toLocaleString('en-IN', { timeZone:
'Asia/Kolkata' });

    doc.text(`Vehicle Number: ${vehicleData.vehicleNumber}`, 10, 10);
    doc.text(`Entry Time: ${date}`, 10, 20);
    doc.text(`Slot Number: ${vehicleData.slot}`, 10, 30);
    doc.save('checkin_invoice.pdf');
  };

  return (
    <div className="max-w-md mx-auto  shadow-md rounded px-8 pt-6 pb-8 mb-4">
    <h2 className="block text-gray-700 text-xl font-bold mb-2">Check In</h2>
    <form onSubmit={handleSubmit} className="mb-4">
      <input
        className="shadow appearance-none border rounded w-full py-2 px-3
text-gray-700 leading-tight focus:outline-none focus:shadow-outline"
        type="text"
        placeholder="Vehicle Number"
        value={vehicleNumber}
        onChange={(e) => setVehicleNumber(e.target.value)}
        required
      />
      <button type="submit" className="bg-blue-500 hover:bg-blue-700 text-white
font-bold py-2 px-4 rounded focus:outline-none focus:shadow-outline mt-4">
        Check In
      </button>
    </form>
    {message && <p className="text-red-500 text-xs italic">{message}</p>}
    {vehicleData && <button onClick={generateInvoice} className="bg-green-500
hover:bg-green-700 text-white font-bold py-2 px-4 rounded focus:outline-none
focus:shadow-outline mt-2">
      Generate Invoice
    </button>}
    </div>
  );
};

export default CheckIn;
```

CHECKOUT.JSX:

```
import { useState } from "react";
import axios from "axios";
import { jsPDF } from "jspdf";

const CheckOut = () => {
  const [vehicleNumber, setVehicleNumber] = useState("");
  const [message, setMessage] = useState("");
  const [vehicleData, setVehicleData] = useState(null);

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await axios.patch(
        "http://localhost:5000/api/vehicles/checkout",
        { vehicleNumber }
      );
      if (response.data) {
        setMessage(
          <div
            className="p-4 mb-4 text-sm text-green-700 bg-green-100 rounded-lg
dark:bg-green-200 dark:text-green-800"
            role="alert"
          >
            <span className="font-medium">Success!</span> Vehicle checked out
            successfully! Fee: ${response.data.fee}.
          </div>
        );
        setVehicleData(response.data);
      } else {
        setMessage("Invalid data received from server");
      }
    } catch (error) {
      console.error("Error checking out vehicle:", error);
      setMessage("Error checking out vehicle");
    }
  };

  const generateInvoice = () => {
    if (!vehicleData) return;

    const doc = new jsPDF();
    doc.text(`Vehicle Number: ${vehicleData.vehicleNumber}`, 10, 10);
    doc.text(`Entry Time: ${vehicleData.entryTime}`, 10, 20);
    doc.text(`Exit Time: ${vehicleData.checkoutTime}`, 10, 30);
    doc.text(`Slot Number: ${vehicleData.slot}`, 10, 40);
    doc.text(`Fee: $${vehicleData.fee}`, 10, 50);
    doc.save("checkout_invoice.pdf");
  };

  return (
    <div className="max-w-md mx-auto bg-white shadow-md rounded px-8 pt-6 pb-8 mb-4">
      <h2 className="block text-gray-700 text-xl font-bold mb-2">Check Out</h2>
      <form onSubmit={handleSubmit} className="mb-4">
        <input
          className="shadow appearance-none border rounded w-full py-2 px-3
text-gray-700 leading-tight focus:outline-none focus:shadow-outline"
          type="text"
          placeholder="Vehicle Number"
          value={vehicleNumber}
          onChange={(e) => setVehicleNumber(e.target.value)}
          required
        />
        <button
          type="submit"
          className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4
rounded focus:outline-none focus:shadow-outline mt-4"
        >
          Check Out
        </button>
      </form>
      {message && <p className="text-red-500 text-xs italic">{message}</p>}
      {vehicleData && (
        <button
          onClick={generateInvoice}
          className="bg-green-500 hover:bg-green-700 text-white font-bold py-2 px-4
rounded focus:outline-none focus:shadow-outline mt-2"
        >
          Generate Invoice
        </button>
      )}
    </div>
  );
};

export default CheckOut;
```

SLOTSTATUS.JSX:

```
import  { useState } from 'react';
import axios from 'axios';

const SlotStatus = () => {
    const [status, setStatus] = useState(null);
    const [error, setError] = useState('');

    const fetchStatus = async () => {
        try {
            const response = await axios.get('http://localhost:5000/api/slots/status');
            setStatus(response.data);
            setError('');
        } catch (error) {
            setError('Error fetching slot status');
        }
    };

    return (
        <div className="max-w-md mx-auto bg-white shadow-md rounded-lg p-6">
            <h2 className="text-xl font-semibold mb-4">Slot Status</h2>
            <button onClick={fetchStatus} className="bg-blue-500 hover:bg-blue-700
text-white font-bold py-2 px-4 rounded">
                Check Slot Status
            </button>
            {error && <p className="text-red-500 text-sm mt-2">{error}</p>}
            {status && (
                <div className="mt-4">
                    <p>Total Slots: <span
className="font-semibold">{status.totalSlots}</span></p>
                    <p>Occupied Slots: <span
className="font-semibold">{status.occupiedSlots}</span></p>
                    <p>Remaining Slots: <span
className="font-semibold">{status.remainingSlots}</span></p>
                </div>
            )}
        </div>
    );
};

export default SlotStatus;
```

FINDVEHICLE.JSX:

```
import { useState } from 'react';
import axios from 'axios';

const FindVehicleSlot = () => {
    const [vehicleNumber, setVehicleNumber] = useState('');
    const [slot, setSlot] = useState(null);
    const [error, setError] = useState('');

    const handleFind = async () => {
        try {
            const response = await
axios.get(`http://localhost:5000/api/slots/find/${vehicleNumber}`);
            setSlot(response.data.slotNumber);
            setError('');
        } catch (error) {
            setError('Vehicle not found');
        }
    };

    return (
        <div className="max-w-md mx-auto bg-white shadow-md rounded px-8 pt-6 pb-8
mb-4">
        <h2 className="block text-gray-700 text-xl font-bold mb-2">Find Vehicle
Slot</h2>
        <input
            className="shadow appearance-none border rounded w-full py-2 px-3
text-gray-700 leading-tight focus:outline-none focus:shadow-outline"
            type="text"
            placeholder="Vehicle Number"
            value={vehicleNumber}
            onChange={(e) => setVehicleNumber(e.target.value)}
            required
        />
        {error && <p className="text-red-500 text-xs italic">{error}</p>}
        <button
            onClick={handleFind}
            className="bg-blue-500 hover:bg-blue-700 text-white font-bold py-2 px-4
rounded focus:outline-none focus:shadow-outline mt-4"
        >
            Find Slot
        </button>
        {slot && <p className="mt-2">Slot Number: <span
className="font-semibold">{slot}</span></p>}
    </div>
);
}

export default FindVehicleSlot;
```

5.2 BACKEND

5.2.1 MODULE:

SLOTMODEL.JS

```
const mongoose = require('mongoose');

const slotSchema = new mongoose.Schema({
    slotNumber: {
        type: Number,
        required: true,
        unique: true,
    },
    isOccupied: {
        type: Boolean,
        default: false,
    },
});

const Slot = mongoose.model('Slot', slotSchema);

module.exports = Slot;
```

VEHICLEMODEL.JS

```
const mongoose = require('mongoose');

const vehicleSchema = new mongoose.Schema({
    vehicleNumber: {
        type: String,
        required: true,
        validate: {
            validator: function(v) {
                return /^[A-Za-z]{2}\d{2}[A-Za-z]{2}\d{4}$/.test(v);
            },
            message: props => `${props.value} is not a valid vehicle number! Format
should be: AA99AA9999`
        }
    },
    entryTime: {
        type: Date,
        default: Date.now,
    },
    checkoutTime: {
        type: Date,
    },
    fee: {
        type: Number,
    },
    slot: {
        type: Number,
    },
});
```

```
const Vehicle = mongoose.model('Vehicle', vehicleSchema);

module.exports = Vehicle;
```

5.2.2 CONTROLLER:

```
const Vehicle = require('../models/vehicleModel');
const Slot = require('../models/slotModel');
const createVehicle = async (req, res) => {
    const { vehicleNumber } = req.body;

    if (!vehicleNumber) {
        return res.status(400).json({ message: 'Vehicle number is required' });
    }

    try {
        const availableSlot = await Slot.findOne({ isOccupied: false });
        if (!availableSlot) {
            return res.status(400).json({ message: 'No available slots' });
        }

        const newVehicle = new Vehicle({ vehicleNumber, slot: availableSlot.slotNumber
});
        await newVehicle.save();

        availableSlot.isOccupied = true;
        await availableSlot.save();

        res.status(201).json(newVehicle);
    } catch (error) {
        res.status(500).json({ message: 'Server Error', error });
    }
};
const checkoutVehicle = async (req, res) => {
    const { vehicleNumber } = req.body;

    if (!vehicleNumber) {
        return res.status(400).json({ message: 'Vehicle number is required' });
    }

    try {
        const vehicle = await Vehicle.findOne({ vehicleNumber });
        if (!vehicle) {
            return res.status(404).json({ message: 'Vehicle not found' });
        }

        if (vehicle.checkoutTime) {
            return res.status(400).json({ message: 'Vehicle has already checked out' });
        }

        vehicle.checkoutTime = Date.now();
        const duration = (vehicle.checkoutTime - vehicle.entryTime) / (1000 * 60 * 60);
        const rate = 10;
        vehicle.fee = Math.ceil(duration) * rate;
        await vehicle.save();
        const slot = await Slot.findOne({ slotNumber: vehicle.slot });
        slot.isOccupied = false;
        slot.vehicleNumber = null;
        await slot.save();
        const entryTimeIST = new Date(vehicle.entryTime).toLocaleString('en-IN', {
timeZone: 'Asia/Kolkata' });
        const checkoutTimeIST = new Date(vehicle.checkoutTime).toLocaleString('en-IN', {
timeZone: 'Asia/Kolkata' });
        const response = {
            vehicleNumber: vehicle.vehicleNumber,
            fee: vehicle.fee,
            slot: vehicle.slot,
            entryTime: entryTimeIST,
            checkoutTime: checkoutTimeIST
        };

        res.status(200).json(response);
    } catch (error) {
        res.status(500).json({ message: 'Server Error', error });
    }
};
module.exports = { createVehicle, checkoutVehicle };
```

5.2.3 ROUTE:

SLOTROUTES.JS

```
const express = require('express');
const router = express.Router();
const Slot = require('../models/slotModel');
const Vehicle = require('../models/vehicleModel');
router.post('/initialize', async (req, res) => {
    const { slotCount } = req.body;
    if (!slotCount) {
        return res.status(400).json({ message: 'Slot count is required' });
    }
    try {
        for (let i = 1; i <= slotCount; i++) {
            const slot = new Slot({ slotNumber: i });
            await slot.save();
        }
        res.status(201).json({ message: 'Slots initialized' });
    } catch (error) {
        res.status(500).json({ message: 'Server Error', error });
    }
});
router.delete('/deleteslots', async (req, res) => {
    try {
        await Slot.deleteMany();
        res.status(200).json({ message: 'All slots deleted successfully' });
    } catch (error) {
        res.status(500).json({ message: 'Server Error', error });
    }
});
```

```
router.get('/status', async (req, res) => {
    try {
        const totalSlots = await Slot.countDocuments();
        const occupiedSlots = await Slot.countDocuments({isOccupied: true });
        const remainingSlots = totalSlots - occupiedSlots;
        res.status(200).json({ totalSlots, occupiedSlots, remainingSlots });
    } catch (error) {
        res.status(500).json({ message: 'Error getting slot status' });
    }
});

router.get('/find/:vehicleNumber', async (req, res) => {
    try {
        const vehicle = await Vehicle.findOne({ vehicleNumber: req.params.vehicleNumber
});
        if (vehicle) {
            const slot = await Slot.findOne({ slotNumber: vehicle.slot, isOccupied: true
});
            if (slot) {
                res.status(200).json({ slotNumber: slot.slotNumber });
            } else {
                res.status(404).json({ message: 'Slot not found or not occupied' });
            }
        } else {
            res.status(404).json({ message: 'Vehicle not found' });
        }
    } catch (error) {
        res.status(500).json({ message: 'Error finding vehicle slot', error });
    }
});



module.exports = router;
```

VEHICLEROUTES.JS

```
const express = require('express');
const { createVehicle, checkoutVehicle } = require('../controllers/vehicleController');
const router = express.Router();

router.post('/', createVehicle);
router.patch('/checkout', checkoutVehicle);

module.exports = router;
```

6. RESULT

HOME

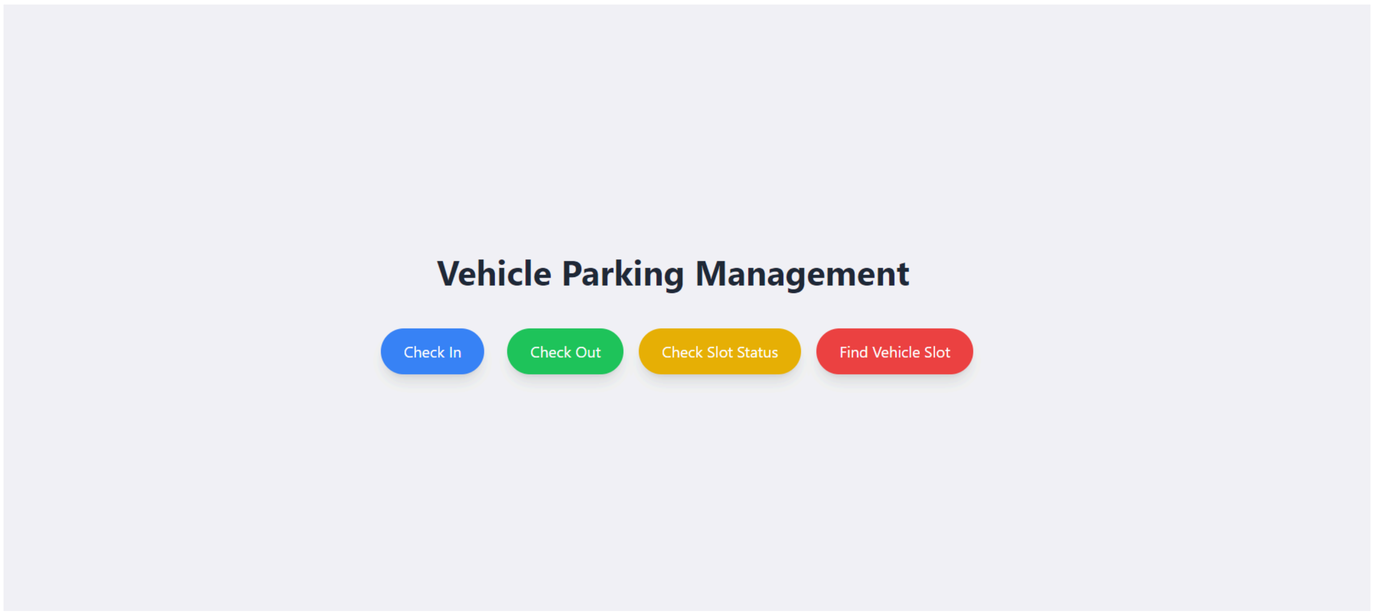
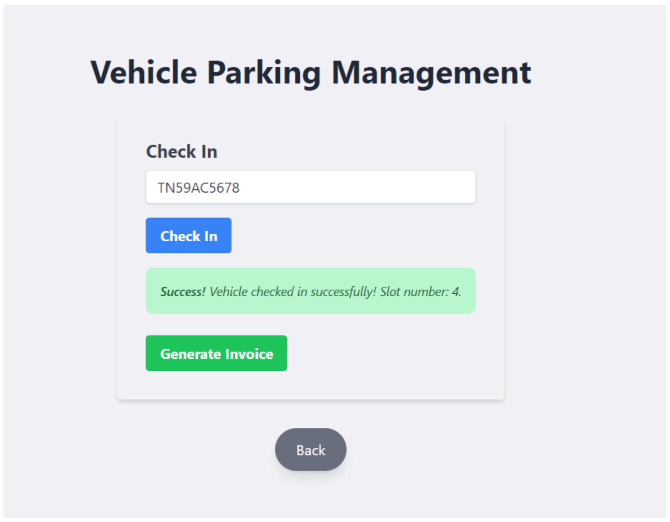


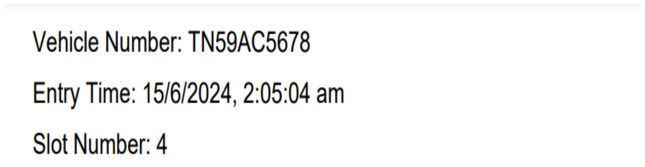
Fig 6.1 Home Page

Home Page: This page shows the various operations that can be performed

CHECK IN:



Success! Vehicle checked in successfully! Slot number: 4.

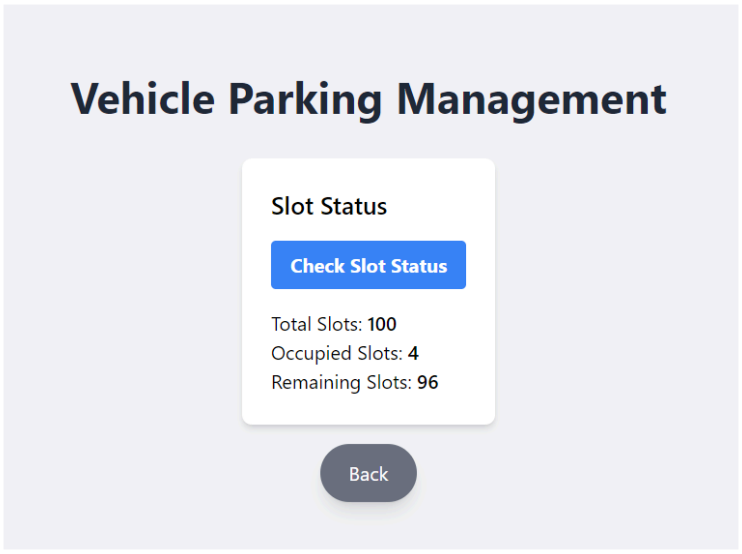


Vehicle Number: TN59AC5678
Entry Time: 15/6/2024, 2:05:04 am
Slot Number: 4

Fig 6.2 Check In **Fig 6.3** Slot allocated

Check In: This page helps to allocate slot to park the vehicle.

SLOT STATUS:

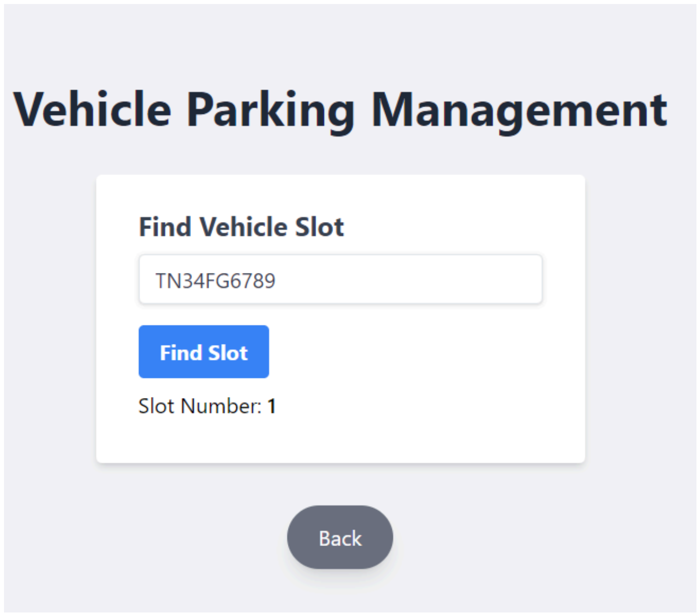


Total Slots: 100
Occupied Slots: 4
Remaining Slots: 96

Fig 6.4 Slot Status

Slot Status: This page helps to find the status of the slots

FETCH SLOT:


Vehicle Parking Management

Find Vehicle Slot

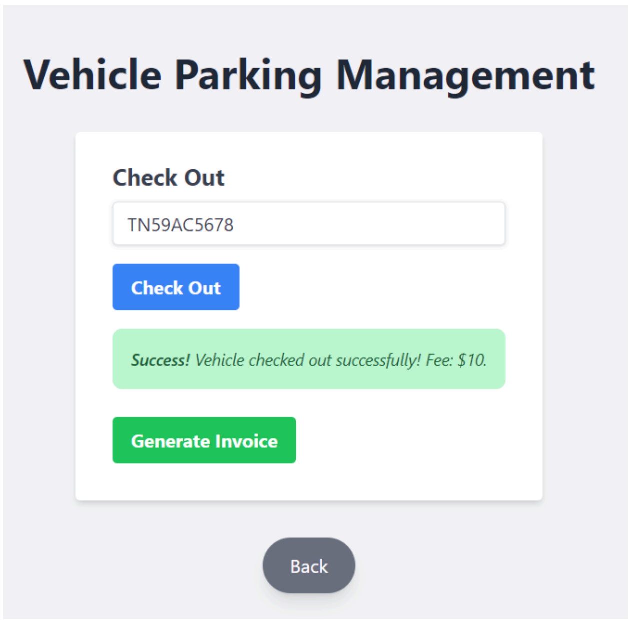
TN34FG6789

Slot Number: **1**

Fig 6.5 Fetch Slot

Fetch Slot: This page helps to fetch the slot of the vehicle.

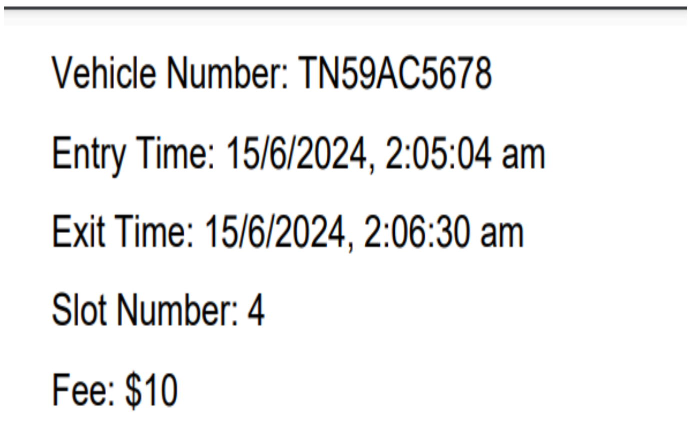
CHECK OUT:


Vehicle Parking Management

Check Out

TN59AC5678

Success! Vehicle checked out successfully! Fee: $10.


Vehicle Number: TN59AC5678

Entry Time: 15/6/2024, 2:05:04 am

Exit Time: 15/6/2024, 2:06:30 am

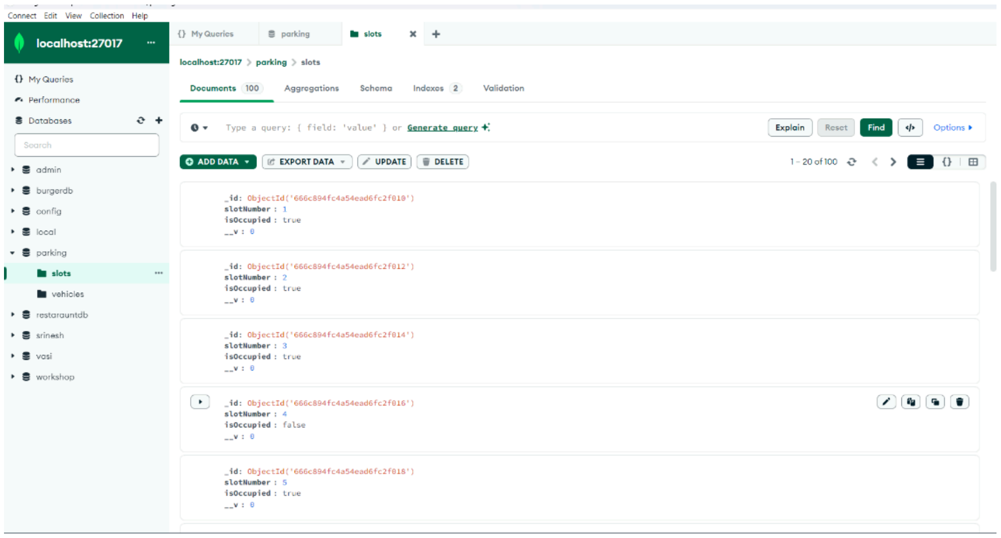
Slot Number: 4

Fee: $10

Fig 6.6 Check Out **Fig 6.7 Report**

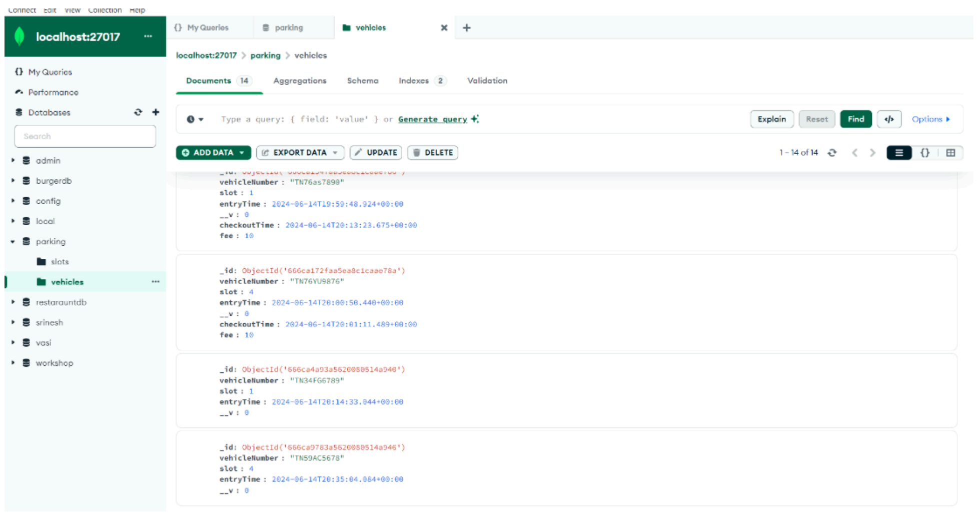
Check Out: This page gives the details of the Vehicle with in time and out time and amount for parking

DATABASE



```
_id: ObjectId('666c894fc4a54ead6fc2f010')
slotNumber : 1
isOccupied : true
__v : 0

_id: ObjectId('666c894fc4a54ead6fc2f012')
slotNumber : 2
isOccupied : true
__v : 0

_id: ObjectId('666c894fc4a54ead6fc2f014')
slotNumber : 3
isOccupied : true
__v : 0

_id: ObjectId('666c894fc4a54ead6fc2f016')
slotNumber : 4
isOccupied : false
__v : 0

_id: ObjectId('666c894fc4a54ead6fc2f018')
slotNumber : 5
isOccupied : true
__v : 0
```

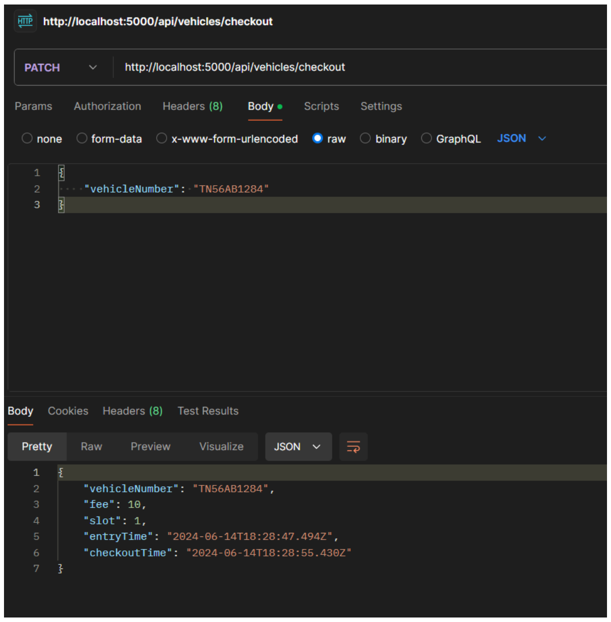
Fig 6.8 Database Check In



```
vehicleNumber : "TN76as7890"
slot : 1
entryTime : 2024-06-14T19:59:48.924+00:00
__v : 0
checkoutTime : 2024-06-14T20:13:23.675+00:00
fee : 10

_id: ObjectId('666ca172faa5ea8c1caae78a')
vehicleNumber : "TN76YU9876"
slot : 4
entryTime : 2024-06-14T20:00:50.440+00:00
__v : 0
checkoutTime : 2024-06-14T20:01:11.489+00:00
fee : 10

_id: ObjectId('666ca4a93a5620080514a940')
vehicleNumber : "TN34FG6789"
slot : 1
entryTime : 2024-06-14T20:14:33.044+00:00
__v : 0

_id: ObjectId('666ca9783a5620080514a946')
vehicleNumber : "TN59AC5678"
slot : 4
entryTime : 2024-06-14T20:35:04.084+00:00
__v : 0
```

Fig 6.9 Database Check Out

POSTMAN



PATCH http://localhost:5000/api/vehicles/checkout

```
{
    "vehicleNumber": "TN56AB1284"
}
```

```
{
    "vehicleNumber": "TN56AB1284",
    "fee": 10,
    "slot": 1,
    "entryTime": "2024-06-14T18:28:47.494Z",
    "checkoutTime": "2024-06-14T18:28:55.430Z"
}
```

Fig 6.10 Postman

CONCLUSION:

The Vehicle Parking System significantly improves parking management by utilizing modern technology for efficient slot allocation, real-time status updates, and automated payment processing. By providing a user-friendly interface and accurate tracking of vehicles, it enhances the overall parking experience for users. The system's ability to manage parking resources effectively reduces congestion and optimizes space utilization. Future enhancements, such as predictive analytics and smart city integration, promise even greater efficiency and convenience. This system represents a substantial advancement in organized and user-friendly parking solution.