

PARKING MANAGEMENT SYSTEM

BY

- **MEGA.U 22BAI1011**
- **RISHIKKESH.R 22BRS1169**
- **JEEVAN.Y 22BRS1161**
- **RAGHUL KRISHNA 22BRS1005**
- **RAHUL.K 22BRS1363**
- **MADHESHWARAN.S 22BRS1183**





- Parking management system project revolutionizes parking by integrating FASTag technology, offering real-time space visualization, and automating vehicle entry, exit, and fee calculation. This user-friendly system tracks parking duration, ensuring a fair pricing structure. It prioritizes scalability and integration, making it adaptable for future advancements and data-driven decision-making. Committed to security and user satisfaction, this project aims to optimize parking facility operations while promoting eco-friendliness and customer convenience.

Introduction



Objective


- The objective of our parking management system project is to design and develop the software to efficiently manage a parking facility. The primary goal is to create a robust, user-friendly and automated system that seamlessly handles vehicle entry and exit, FASTag scanning, parking space allocation, in-time and out-time recording and the calculation of parking fees.

Problem Statement


In urban environments, the traditional methods of managing parking facilities are struggling to cope with the escalating challenges posed by the increasing number of vehicles. Conventional parking systems often lack the efficiency, automation, and integration required to meet the demands of modern cities. The absence of advanced technologies results in congestion, inefficient space utilization, and manual processes that contribute to delays and user dissatisfaction.

Furthermore, the absence of a standardized and automated fee calculation system often leads to inconsistencies and disputes, impacting both operators and users. The lack of real-time visibility into parking space availability exacerbates the problem, causing frustration for drivers seeking parking and inefficiencies in facility operations.


In response to these challenges, our parking management system project seeks to address the following issues:




Inefficient Entry and Exit Processes: The existing manual methods of handling vehicle entry and exit contribute to congestion, delays, and increased carbon emissions. There is a need for an automated system that streamlines these processes for a more seamless and eco-friendly experience.




Outdated Fee Calculation Systems: The absence of a standardized and automated fee calculation mechanism results in inconsistencies, disputes, and a lack of transparency in the pricing structure. Implementing an accurate and automated fee calculation system is essential for fair and efficient financial transactions.




Lack of Real-Time Space Visualization: The inability to provide real-time information on available parking spaces hinders drivers in making informed decisions, leading to increased search times and frustration. A solution that offers dynamic space visualization is critical for optimizing space utilization and improving user experience.



Limited Integration with Modern Technologies: The current parking management systems often lack integration with emerging technologies, hindering adaptability to advancements such as FASTag technology. There is a need for a system that can seamlessly integrate with existing and future technologies for enhanced functionality and scalability.



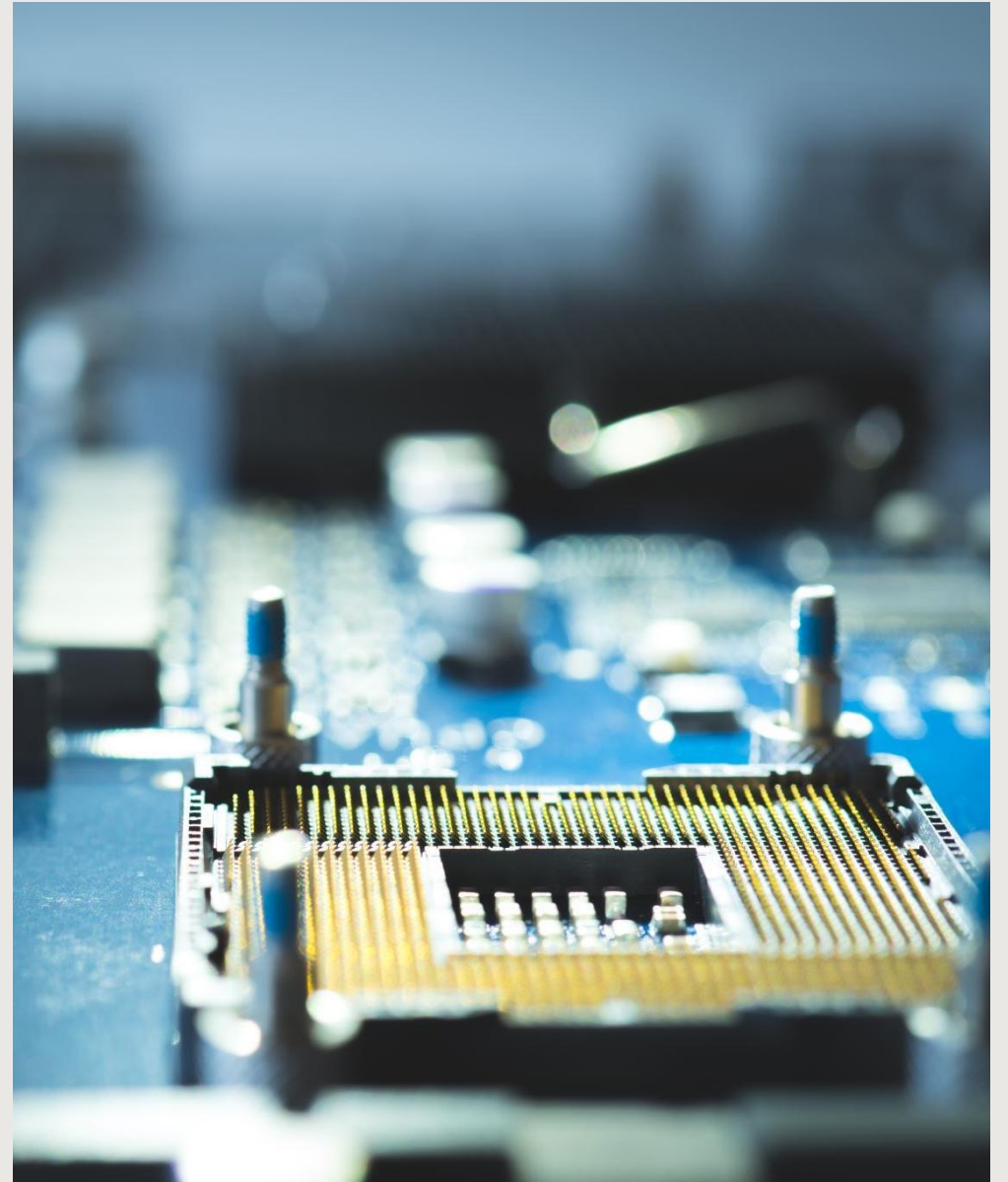
Insufficient Scalability and Adaptability: Many existing parking management systems are rigid and lack scalability, making it challenging to accommodate the dynamic growth of urban areas and the increasing number of vehicles. Our project aims to create a solution that is scalable and adaptable to future technological developments and changing urban landscapes.



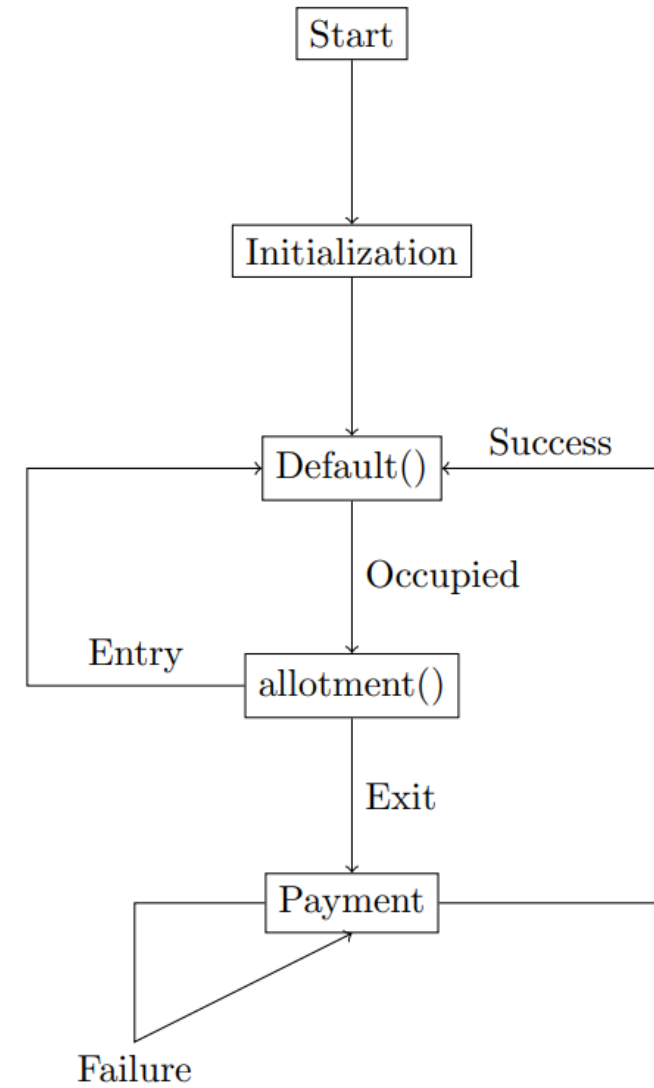
Addressing these challenges will contribute to the creation of a robust, user-friendly, and technologically advanced parking management system that optimizes operations, promotes sustainability, and enhances user satisfaction

Requirements (Software & Hardware)

- **HARDWARE REQUIREMENTS: (MINIMUM)**
- CPU: Any Processor with dual cores or above can handle an Algorithm Visualiser (all models of Intel i3, 17, i9 and any Ryzen processor)
- RAM: At least 4 GB of RAM is recommended. This amount of RAM will handle most visualizations and keep the software responsive.
- Graphics: A basic integrated graphics card such as Intel Iris XE is sufficient for rendering the visualizations created using Seaborn and Celluloid.
- Internet Connection: A stable broadband internet connection is required to access and interact with the web-based Algorithm Visualizer.
- **SOFTWARE REQUIREMENTS: ANY BROWSER**



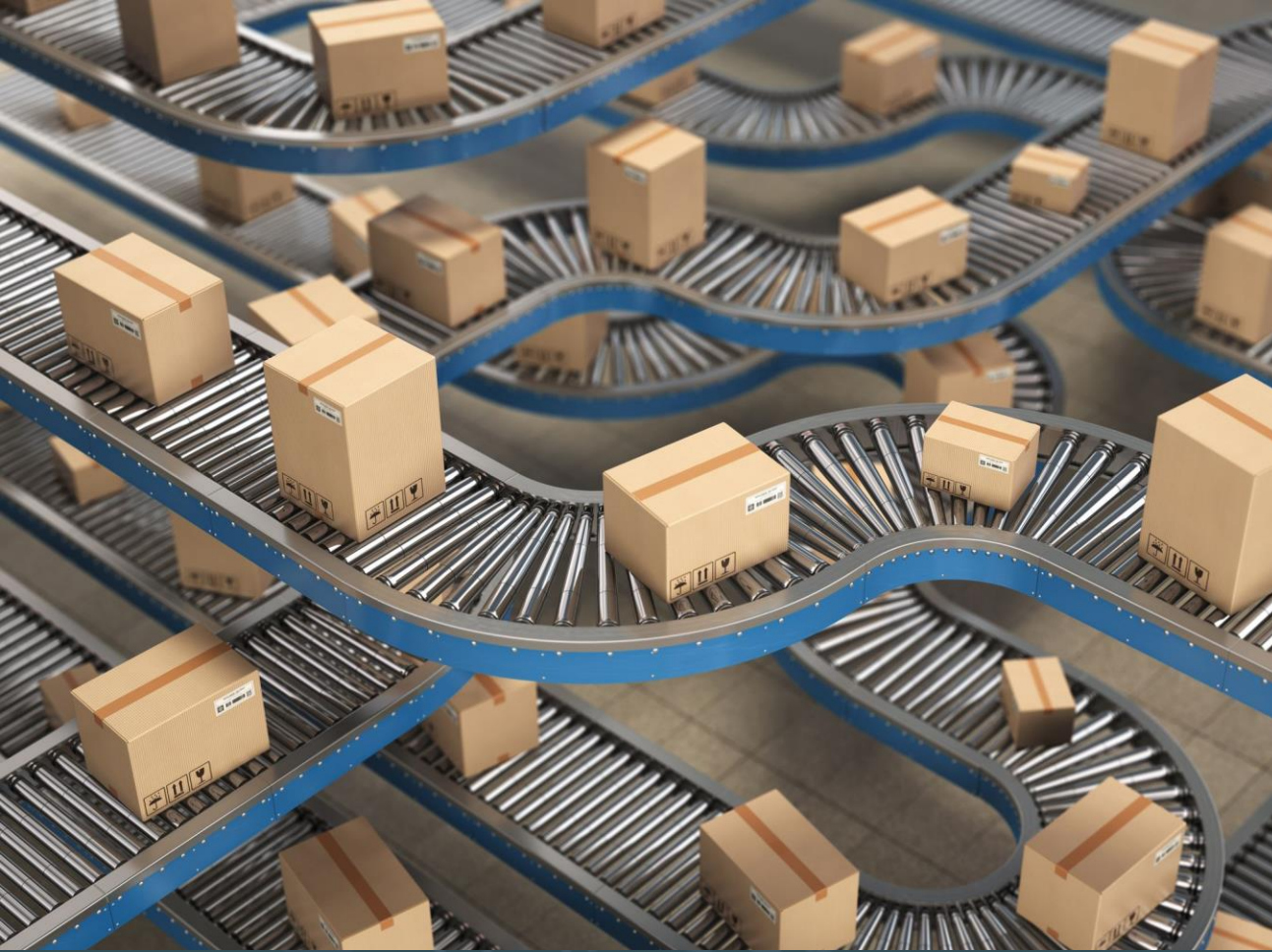
Flow Chart





- FASTag Integration: Integrate FASTag technology for quick and accurate vehicle identification and retrieval of vehicle details.
- Space Visualization: Develop a visual representation of the parking area with small boxes where each box represents a parking space. Utilize grey to indicate occupied spaces and white to indicate available spaces.
- Vehicle Entry: Implement a module to record the entry of vehicles, assign available parking spaces, and change the status of the corresponding box to green.
- In-Time Recording: Automatically record the entry time when a vehicle enters the parking facility.
- Vehicle Exit: Create a process to manage vehicle exits, including updating parking space status and calculating parking fees.
- Out-Time Recording: Automatically record the exit time when a vehicle leaves the parking facility.

Explanation



- Fare Calculation: Implement a fare calculation mechanism based on the duration of the vehicle's stay and any applicable fee structures.
- Data Storage and Management: Develop a database system to store and manage vehicle records, parking space statuses, and fee-related information.
- Efficiency and Scalability: Ensure that the system is designed for efficiency, can handle a large volume of vehicles, and is scalable for future expansion or integration with external systems.
- Security: Implement security measures to protect the system from unauthorized access or data breaches.
- Documentation: Create comprehensive documentation that includes system architecture, database schema, and code documentation to support future development and maintenance.
- Testing and Validation: Conduct rigorous testing and validation to ensure the accuracy and reliability of the system.

Implementation - Modules

Tkinter : Tkinter is Python's standard GUI (Graphical User Interface) toolkit. It provides a set of tools for creating desktop applications with a graphical interface, making it easy to design and implement user-friendly programs.

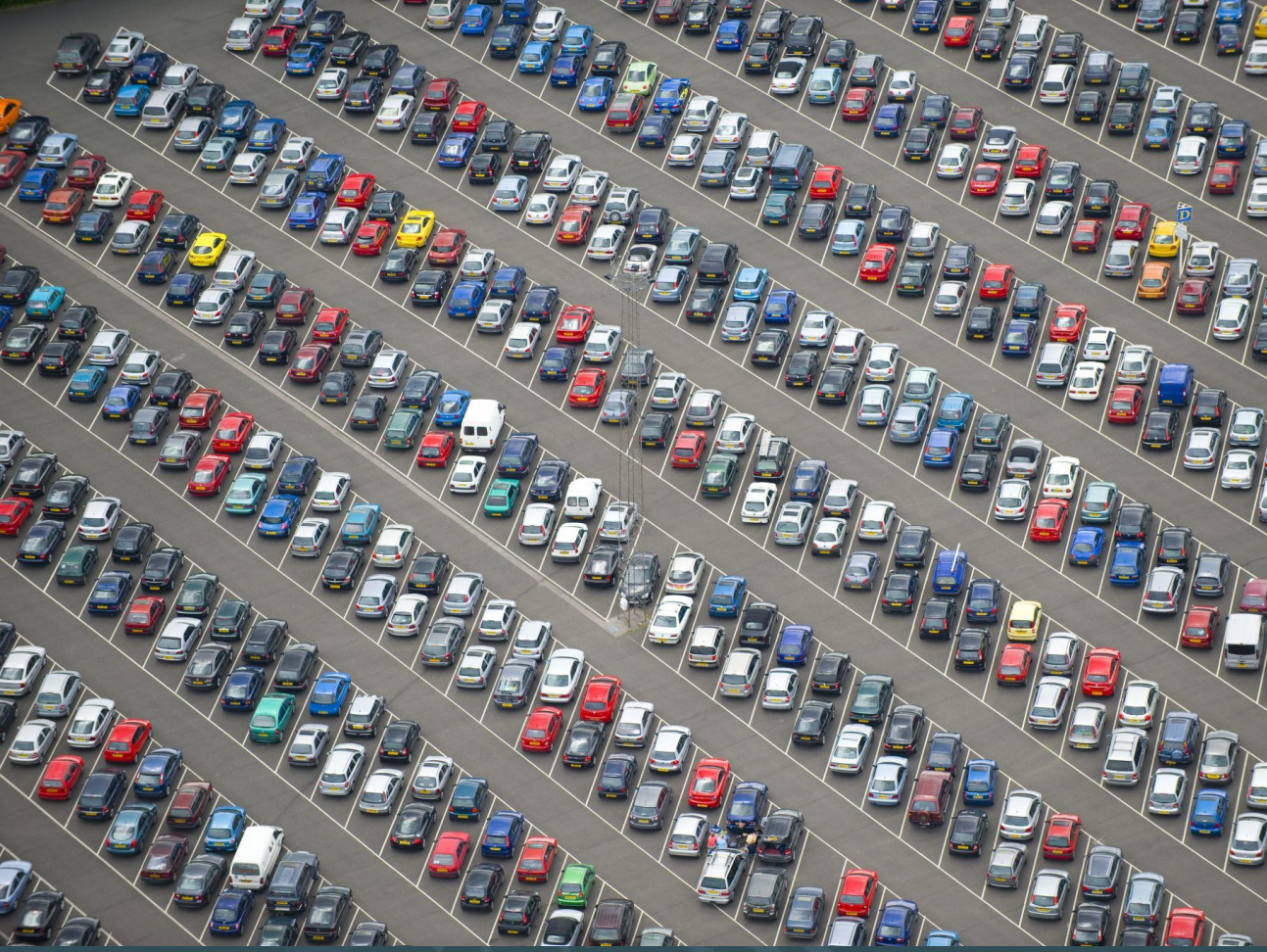
Webbrowser: The webbrowser module allows Python scripts to display web-based documents to users. It provides a high-level interface to allow displaying Web-based documents to users.

PIL (Pillow): PIL (Python Imaging Library), now known as Pillow, is a powerful library for opening, manipulating, and saving many different image file formats. It supports tasks like image cropping, resizing, and basic image processing.

Implementation - Modules

CSV: The CSV module in Python provides functionality to both read from and write to CSV files. It simplifies the process of working with comma-separated values, making it easy to handle tabular data.

Datetime: The datetime module supplies classes for working with dates and times in Python. It allows for creating, formatting, and performing operations on dates and times, providing a versatile tool for time-related calculations and manipulations.



- In this parking management system project, a graphical user interface (GUI) is implemented using the Tkinter library in Python. The system allows cars to enter and exit a simulated parking lot, recording entry and exit times, calculating parking duration and cost. The program reads car details from a CSV file and displays them in the GUI along with a visual representation of the parking lot. The parking lot grid is created using images to represent cars, roads, and parking spots. The system includes functionality for opening a web page when a car exits for payment.

Result and Discussion

Conclusion

- In conclusion, this parking management system project has successfully implemented a graphical user interface using the Tkinter library in Python. The system effectively manages the entry and exit of cars, calculates parking duration and cost, and displays car details within a simulated parking lot environment. The integration of image processing with PIL adds a visual appeal to the user interface. The project serves as a solid starting point for a functional parking management system. Further development opportunities may involve refining code organization, incorporating additional features, and enhancing user interaction to create a more comprehensive and user-friendly solution. Overall, the project lays a foundation for a practical and visually engaging parking management system.

Future Enhancements

- Future Enhancement Of Project The smart parking industry continues to evolve as an increasing number of cities struggle with traffic congestion and inadequate parking availability. While the deployment of sensor technologies continues to be core to the development of smart parking, a wide variety of other technology innovations are also enabling more adaptable systems – including cameras, wireless communications, data analytics, induction loops, smart parking meters, and advanced algorithms. The future of the smart parking market is expected to be significantly influenced by the arrival of automated vehicles (AVs). Several cities around the world are already beginning to trial self-parking vehicles, specialized AV parking lots, and robotic parking valets



- [Geeks for Geeks](#)
- [Tkinter Documentation](#)
- [Programmiz](#)

Bibliography

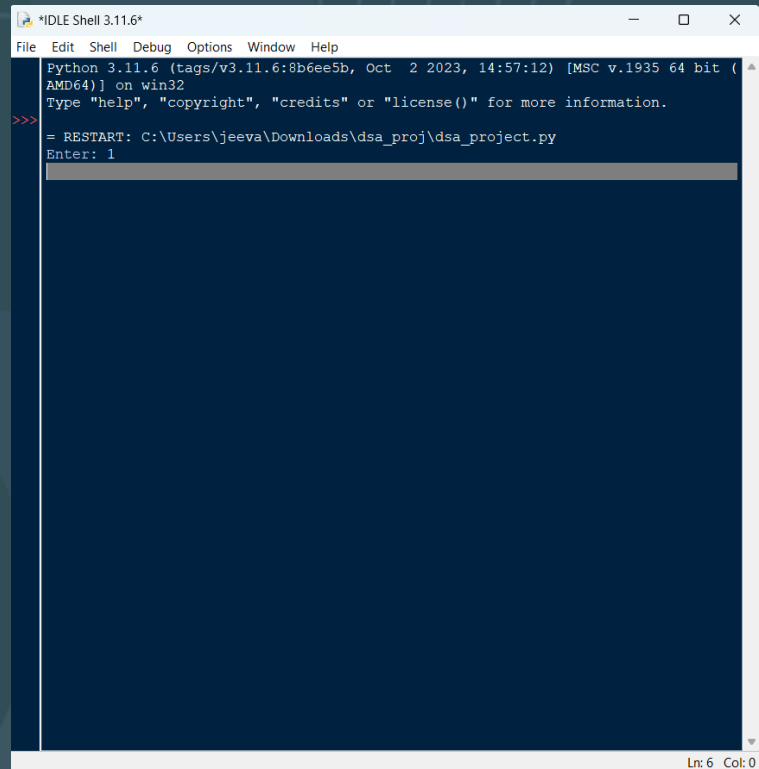


CODE

https://jeevanyoganand.github.io/dsa_proj



Screenshots



```
*IDLE Shell 3.11.6*
File Edit Shell Debug Options Window Help
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\jeeva\Downloads\dsa_proj\dsa_project.py
Enter: 1
```

CAR DETAILS:

=====

CAR BRAND : Toyota

MODEL : Innova

NUMBER PLATE : TN 20 AS 2341

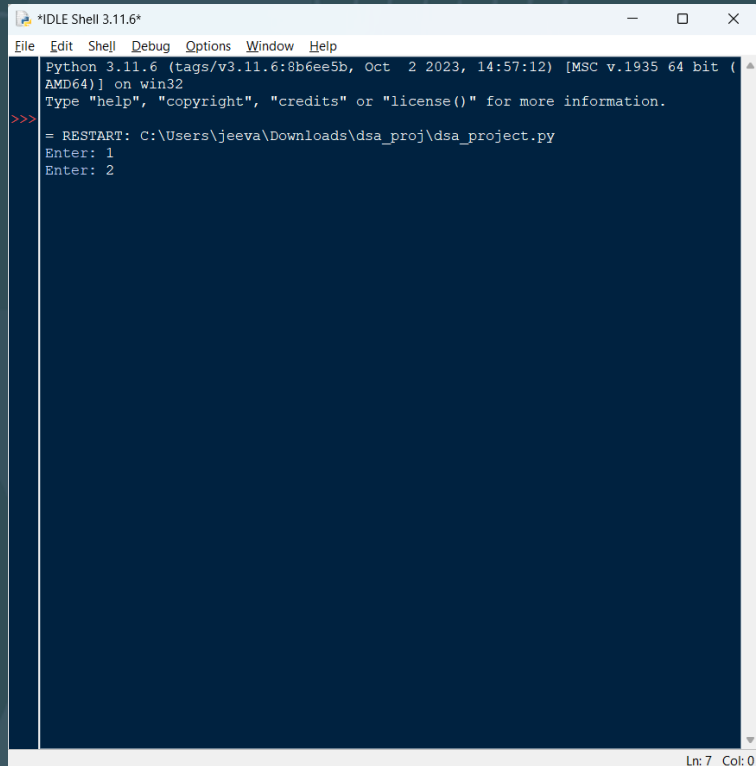
FUEL TYPE : Diesel

=====

ENTRY TIME : 23:45:12

Collect Receipt

Screenshots



```
*IDLE Shell 3.11.6*
File Edit Shell Debug Options Window Help
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\jeeva\Downloads\dsa_proj\dsa_project.py
Enter: 1
Enter: 2
```

Ln: 7 Col: 0

CAR DETAILS:

CAR BRAND : Mercedes-Benz

MODEL : GLS-AMG

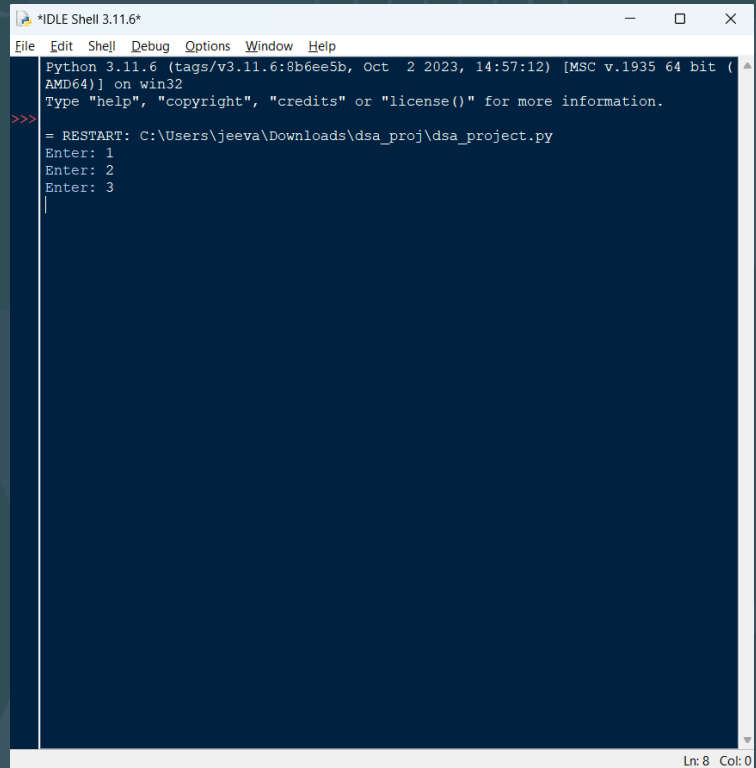
NUMBER PLATE : TN 02 JY 0001

FUEL TYPE : Petrol

ENTRY TIME : 23:45:57

Collect Receipt

Screenshots



```
*IDLE Shell 3.11.6*
File Edit Shell Debug Options Window Help
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\jeeva\Downloads\dsa_proj\dsa_project.py
Enter: 1
Enter: 2
Enter: 3
|
```

Ln: 8 Col: 0

CAR DETAILS:

CAR BRAND : Rolls-Royce

MODEL : Boat_Tail

NUMBER PLATE : TN 10 RR 0009

FUEL TYPE : Petrol

ENTRY TIME : 23:50:10

Collect Receipt

Screenshots

```
*IDLE Shell 3.11.6*
File Edit Shell Debug Options Window Help
Python 3.11.6 (tags/v3.11.6:8b6ee5b, Oct 2 2023, 14:57:12) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\jeeva\Downloads\dsa_proj\dsa_project.py
Enter: 1
Enter: 2
Enter: 3
Enter: 2
|
```

CAR DETAILS:

CAR BRAND : Mercedes-Benz

MODEL : GLS-AMG

NUMBER PLATE : TN 02 JY 0001

FUEL TYPE : Petrol

ENTRY TIME : 23:51:18

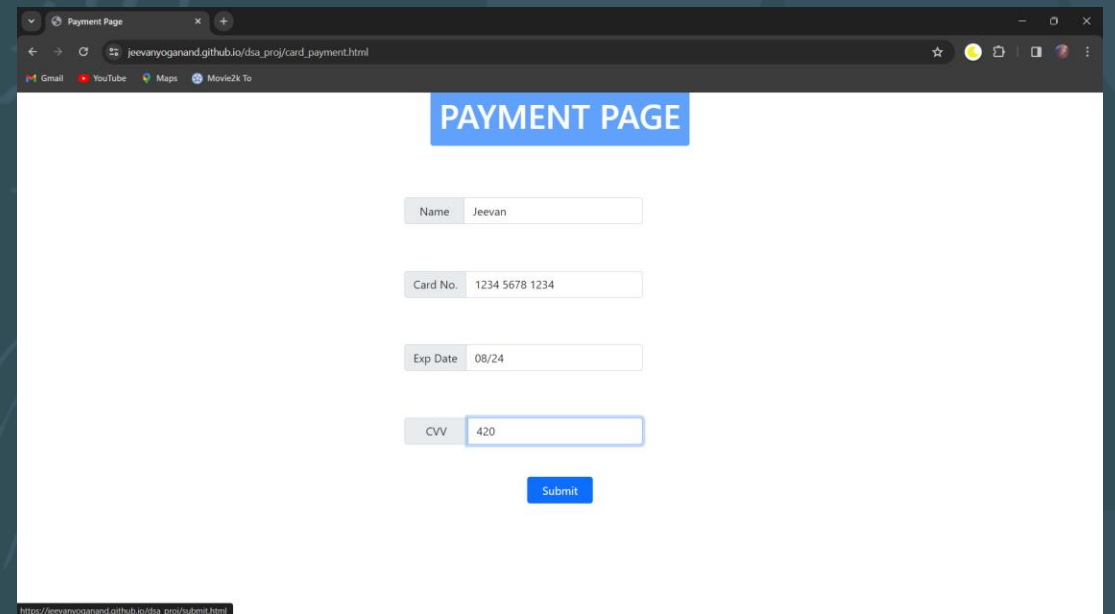
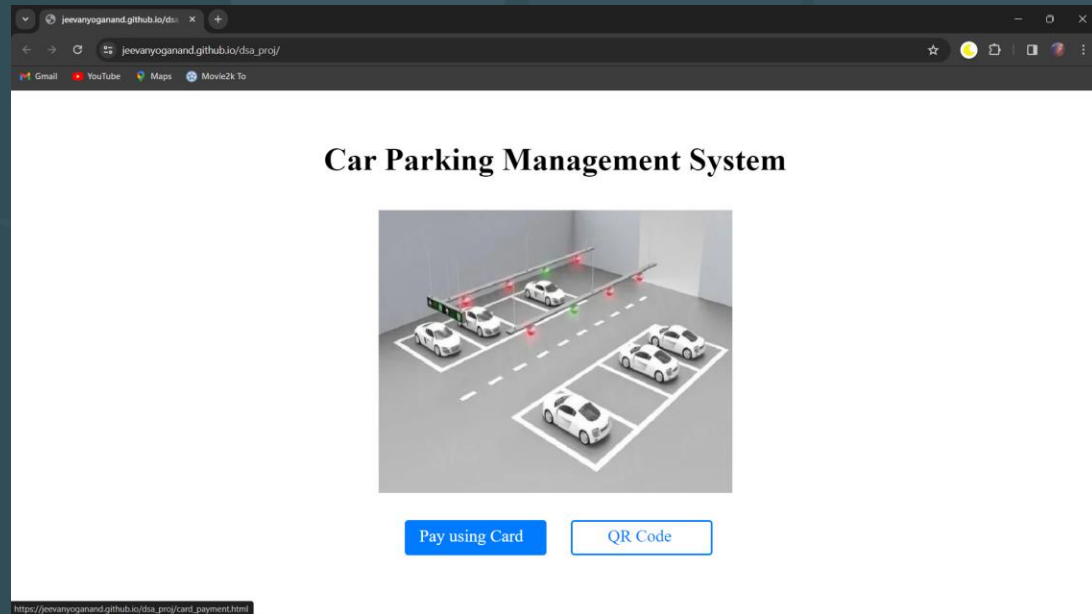
EXIT TIME : 23:47:51

DURATION : 3 mins 26 secs

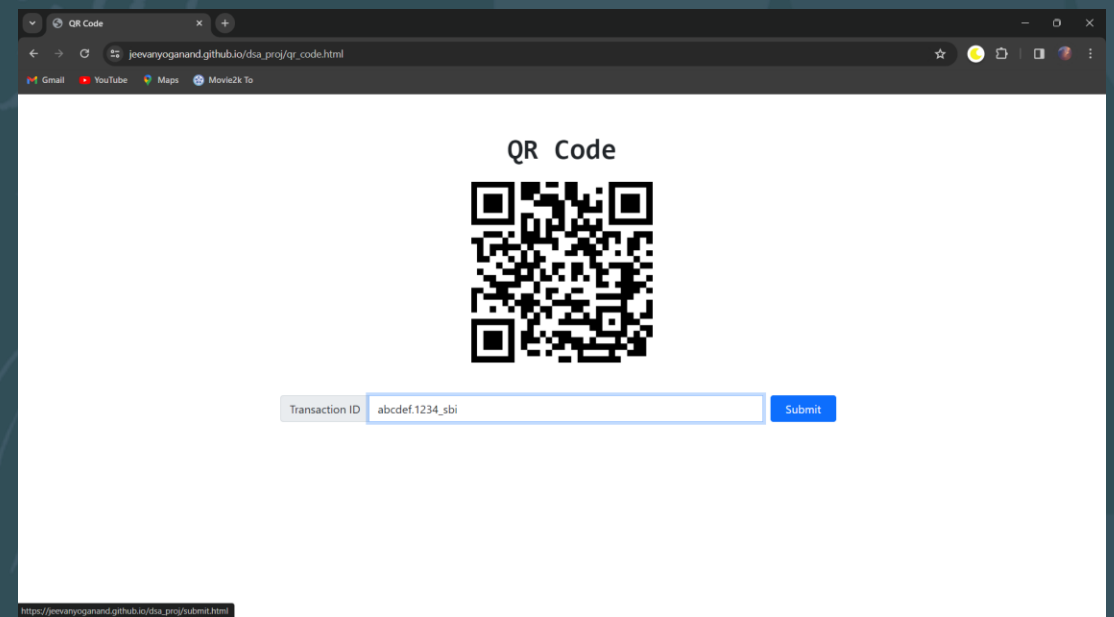
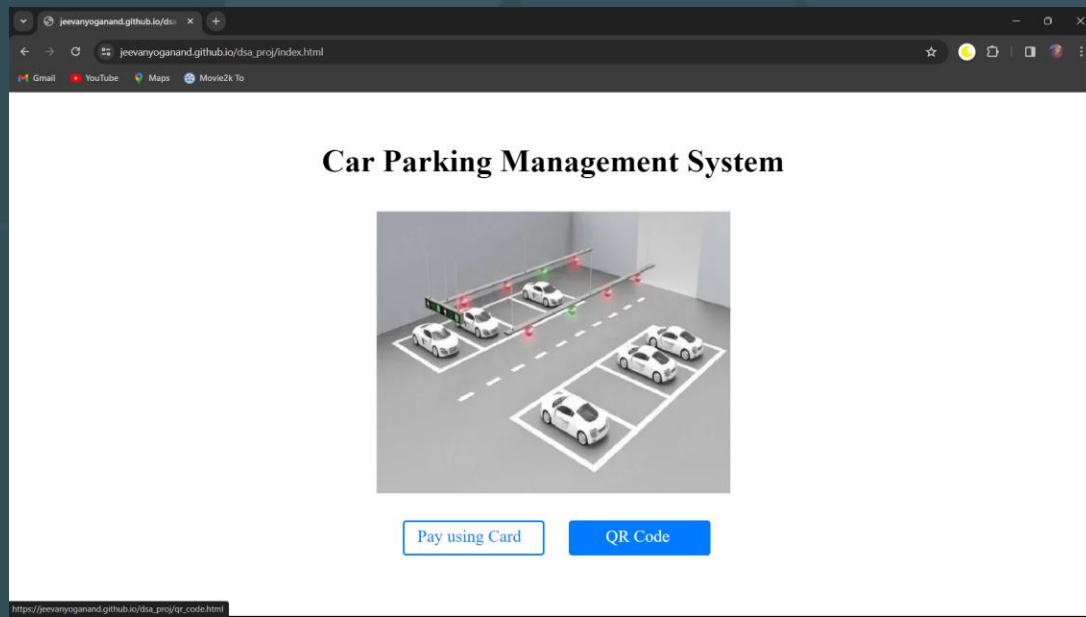
AMOUNT : Rs.13.0

Collect Receipt

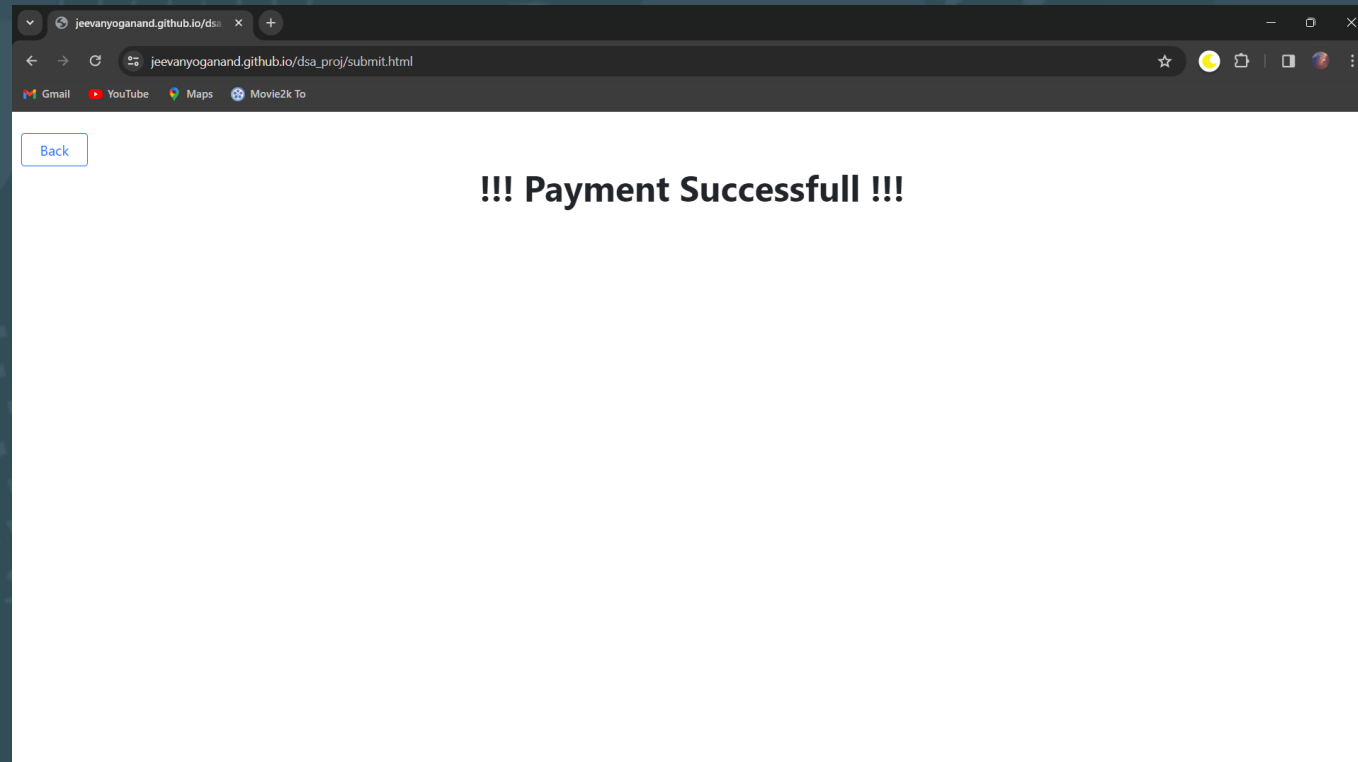
Screenshots



Screenshots



Screenshots





THANK YOU!!

