**EXPERIMENT-6**: To develop Software Requirement Specification (SRS) for taken up project

**CONTENTS TABLE**

1. Introduction

a. Background

b. Overall Description

c. Environmental Characteristics

1. Hardware
2. Peripherals
3. People

d. Interfaces

1. Interface with OS
2. Interface with Database
3. Interface with User

e. Constraints

2. Functional Requirements

3. Non-Functional Requirements

4. Design Diagrams (Use Case, ER, Sequence)

5. Miscellaneous

1. **Introduction**
2. **Background:** The idea behind our project is to upgrade the parking facilities which are not being paid attention to in our country. So we have developed an idea with a 30 minute time constraint to park the vehicle which will save everyone’s time providing a hassle free parking.
3. **Overall Description**: The system automates the process of assigning parking spaces to vehicles entering a facility, tracking parking duration, and calculating parking fee. It is designed for a parking facility with a limited number of spaces.
4. **Environmental Characteristics:**

* **Hardware:** The system primarily relies on computational hardware capable of running the software efficiently, including standard desktops, laptops, or servers.
* **Peripherals:** Input interfaces may include keyboards or mouse for user interaction. Output interfaces could be display screens or printers for presenting the allotted parking space and for the generated fee.
* **People:** The intended user could be anyone entering the facility with our parking system installed.

1. **Interfaces:**
2. **Interface with OS:**

- Runs on standard Windows operating systems and also MacOS.

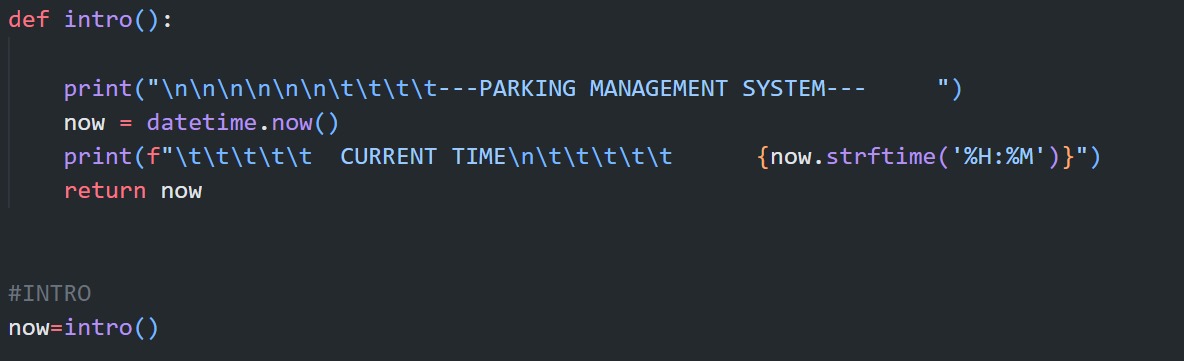
- Accesses system resources like storage, memory, and network connectivity.

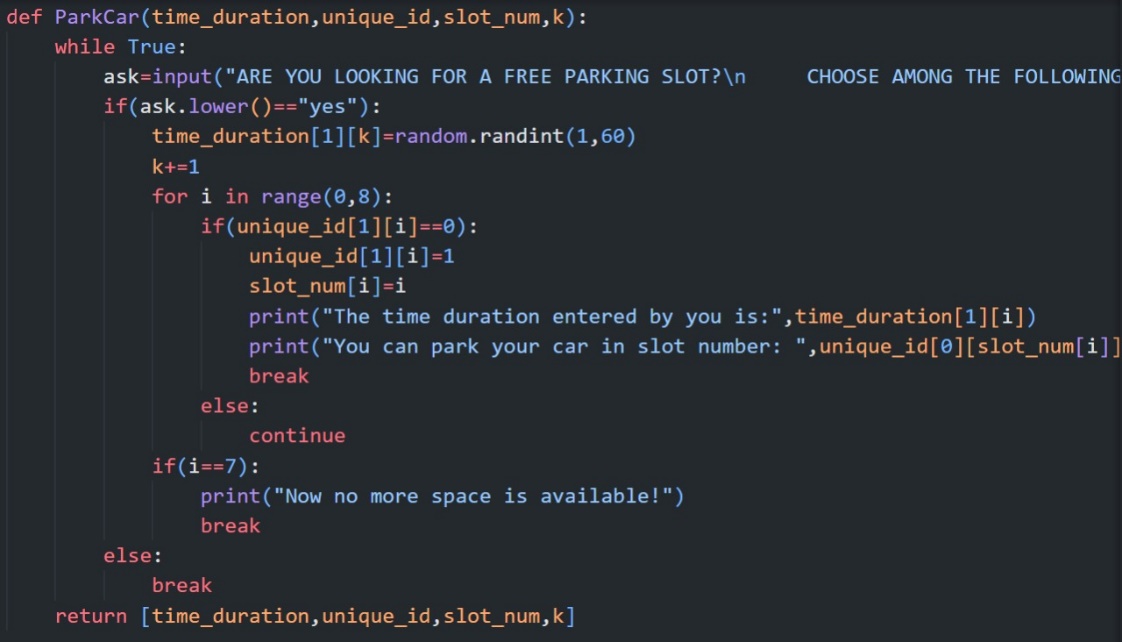
- Following are the screenshots taken from the code written and the corresponding output of the given function.

WINDOWS OPERATING SYSTEMS

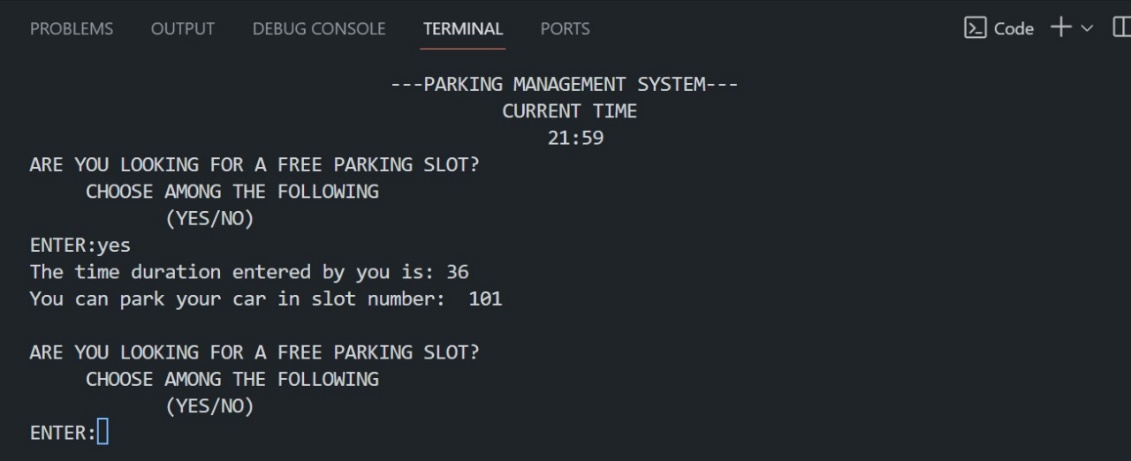
1. Introduction to Parking Management System and asking the user to park the car or not

Code:



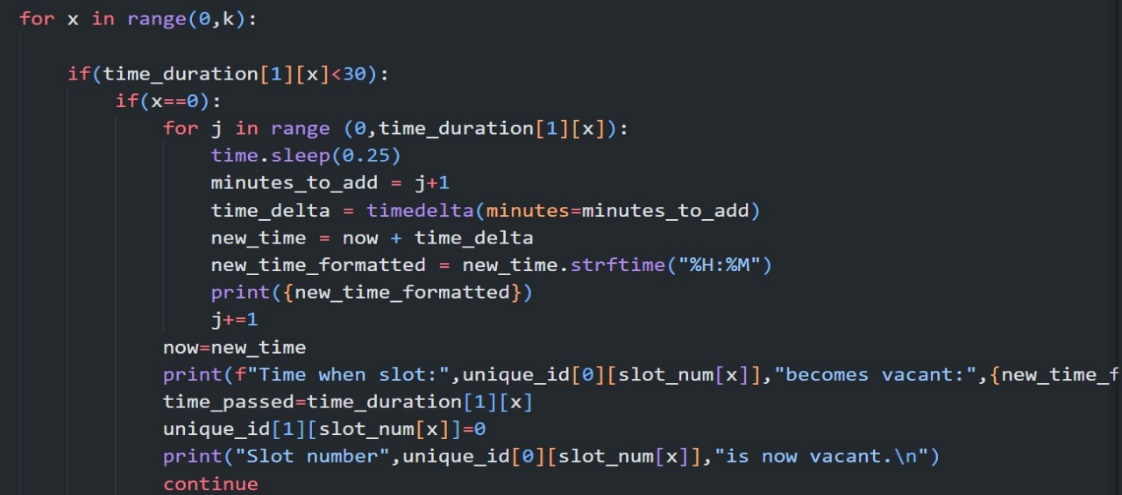


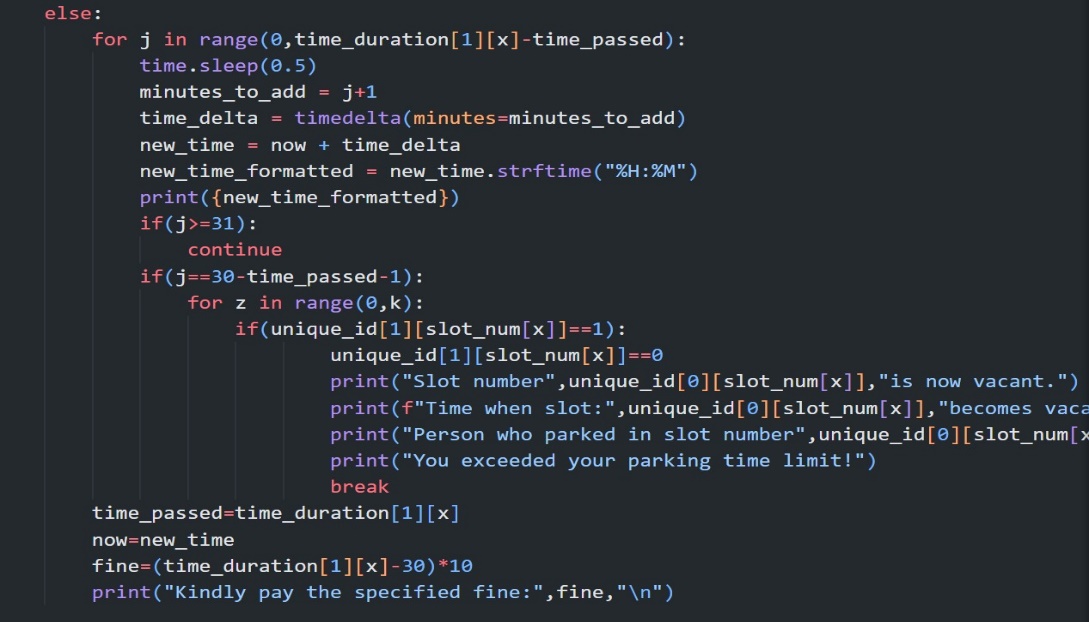
Output:



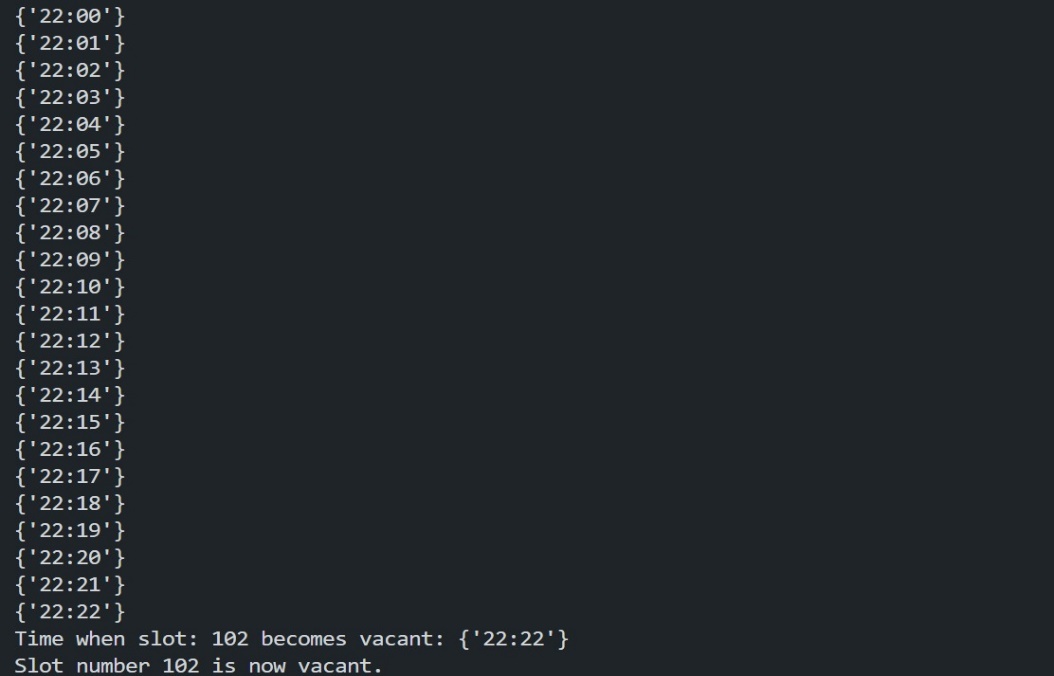
1. Starting the timer and generating the fine

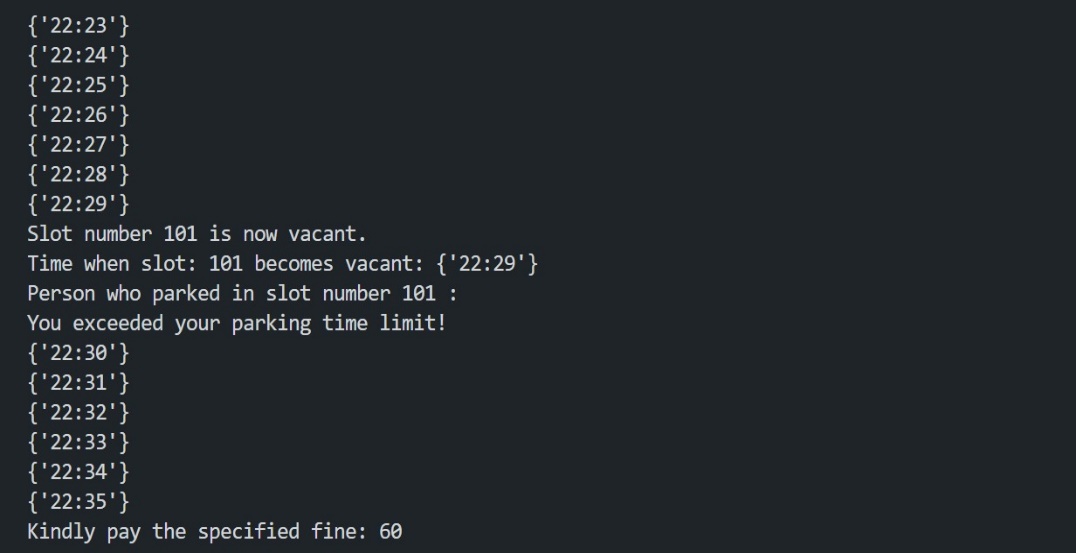
Code:





Output:





1. **Interface with Database:**

- Stores parking space information (location, availability).

- Store fine information.

- Store entry/exit time.

1. **Interface with User:**

- Command Line interface (CLI) for allotting the space.

1. **Constraints:**

* **Limited Parking Spaces:** The system seems designed for a parking facility with a limited number of spaces. The flowchart doesn't specify a way to handle situations where more vehicles arrive than there are available spaces.
* **30-Minute Parking Only:** The system allows only 30 minute of parking which can be less when considered for real world scenarios.

1. **Functional Requirements:**

* **Vehicle Entry Registration**

The system shall generate a unique ticket or identifier for each registered vehicle.

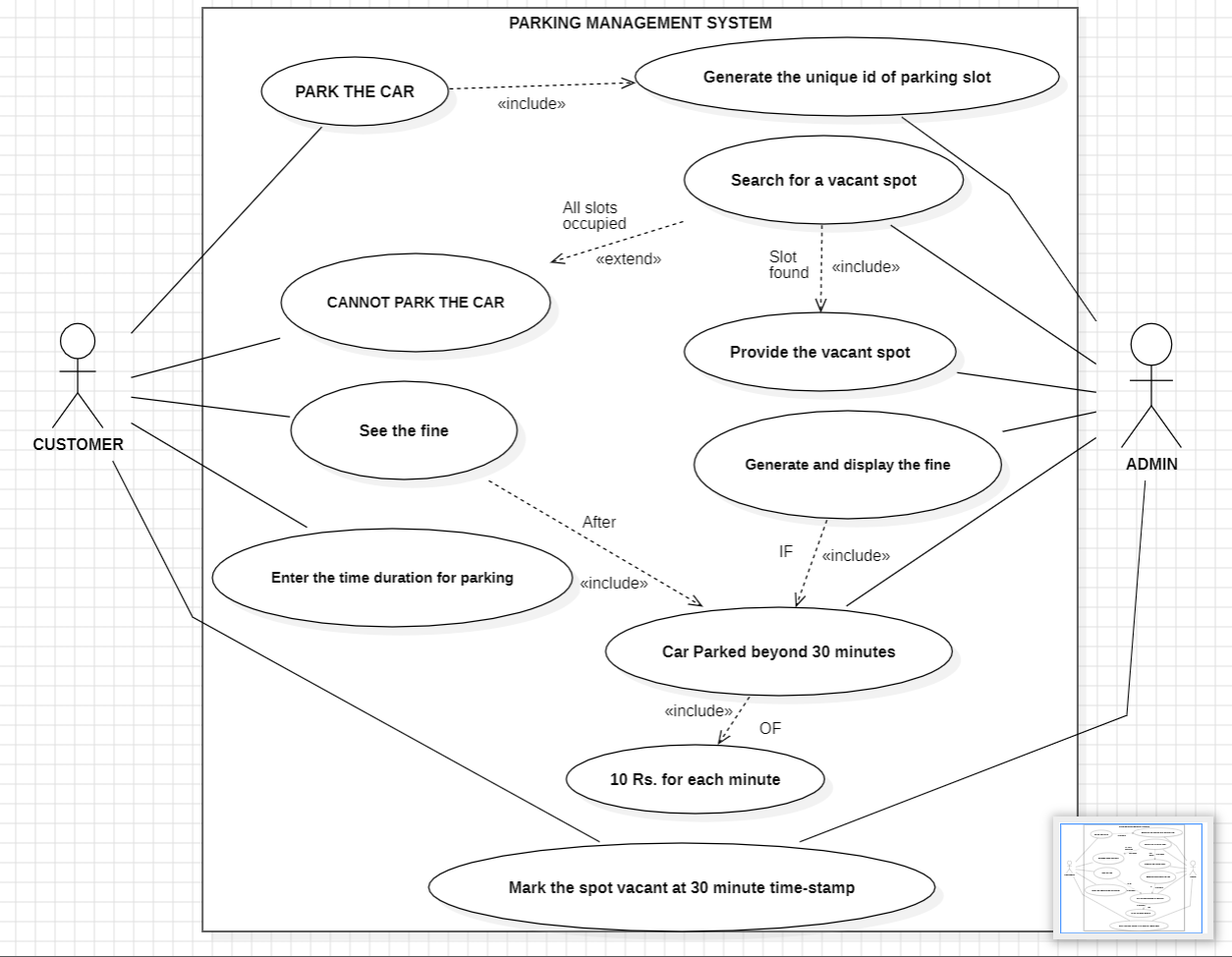
* **Parking Space Allocation and Management**
* Upon vehicle entry registration, the system shall allocate an available parking space based on availability.
* The system shall update the parking space availability status upon vehicle entry and exit.
* **Fee Calculation**
* The system shall calculate parking fees based on factors such as parking duration.
* The system shall issue receipts.
* **Monitoring and Reporting**
* The system shall provide real-time monitoring of parking lot occupancy and availability.
* Administrators shall have access to reports and analytics regarding parking lot utilization and revenue generation.
* **User Authentication and Access Control**
* The system shall authenticate users such as parking attendants, administrators, and customers.
* Access privileges shall be assigned based on user roles, allowing restricted access to sensitive functionalities.

1. **Non-Functional Requirements:**

* **Performance**
* The system shall be capable of handling concurrent user interactions efficiently without significant performance degradation.
* Response times for operations such as vehicle registration, space allocation, and payment processing shall be minimal.
* **Reliability**
* The system shall ensure data integrity and accuracy throughout all operations.
* It should have mechanisms in place to handle system failures gracefully, minimizing data loss and downtime.
* **Security**
* The system shall implement robust security measures to protect sensitive user data and fine information.
* User authentication mechanisms shall prevent unauthorized access to the system functionalities.
* **Usability**
* The system shall have an intuitive user interface that is easy to navigate for both parking attendants and customers.
* Clear instructions and prompts shall be provided to guide users through the parking process.

1. **Design Diagrams:**

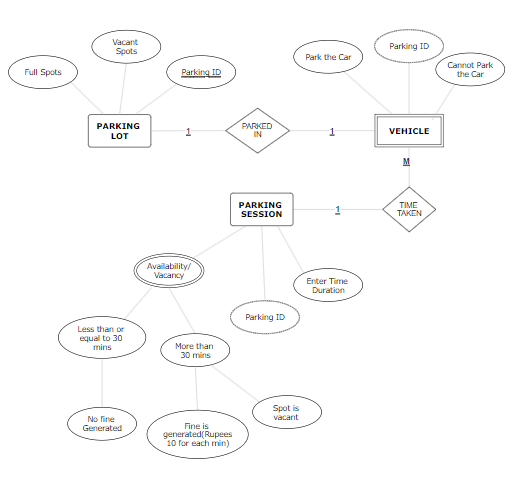
* **Use Case Diagram:**



* **Actors:**
  + **Customer:** The primary user who wants to park their vehicle.
  + **Admin:** Likely responsible for setting fines, managing the system, and potentially resolving issues that arise.
* **Use Cases:**
  + **Generate the Unique ID of Parking Slot:** Ensures each slot has a distinct identifier for tracking.
  + **Search for a Vacant Spot**: Helps the customer locate an available parking space.
  + **Park the Car:** Once the vacant spot has been found, user is provided with a unique id of the spot at which the customer can park the car.

* + **See the Fine:** Allows the customer to view any fines associated with their parking.
  + **Enter the time duration for parking**: The customer inputs their intended parking duration.
  + **Mark the spot vacant at 30-minute timestamp:** The spot frees up once the time duration has been exceeded and the system updates the spot to be shown vacant.
  + **Generate and display the fine:** Generates the fine if the car has been parked beyond 30 minutes of Rs.10 for each minute.
* **ER Diagram:**

This is an **entity-relationship diagram (ERD)** for a parking lot database. It shows the different entities in the database (Parking Lot, Parking Session, Vehicle, Parking ID, Time Taken, Availability/Vacancy, Fine Generated, Spot Vacant) and the relationships between them.



* **The relationship between the three different entities: Parking Lot, Vehicle and Parking Session is described below:**

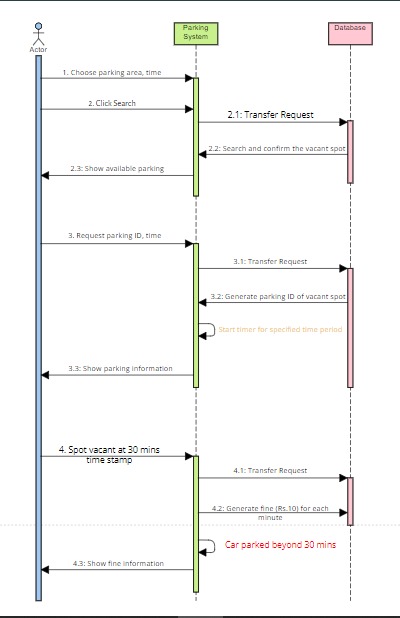
**The Parking Lot entity has a one-to-many relationship with the Parking Session entity** which means that each parking lot can have many parking sessions, but each parking session can only belong to one parking lot. **The Parking Session entity has a one-to-many relationship with the Vehicle entity** which demonstrates that each parking session can only have one vehicle, but each vehicle can have many parking sessions. **The Parking ID entity has a one-to-many relationship with the Parking Session entity**, meaning that each parking ID can only belong to one parking session, but each parking session can have one parking ID. The Time Taken entity has a one-to-many relationship with the Parking Session entity, meaning that each time taken can only belong to one parking session, but each parking session can have one time taken. The Availability/Vacancy entity has a one-to-many relationship with the Parking Session entity, meaning that each availability/vacancy can only belong to one parking session, but each parking session can have one availability/vacancy. The Fine Generated entity has a one-to-many relationship with the Parking Session entity, meaning that each fine generated can only belong to one parking session, but each parking session can have one fine generated. The Spot Vacant entity has a one-to-many relationship with the Parking Session entity, meaning that each spot vacant can only belong to one parking session, but each parking session can have one spot vacant.

* **The explanation of the attributes of the entities:**

The diagram also shows the attributes of each entity. The Parking Lot entity has the following attributes: parking lot ID, parking lot name, and address. The Parking Session entity has the following attributes: parking session ID, parking lot ID, vehicle ID, parked in, park the car, parking ID, cannot park the car, vehicle type, parking session start time, parking session end time, and total time parked. The Vehicle entity has the following attributes: vehicle ID, vehicle type, vehicle make, vehicle model, and vehicle license plate number. The Parking ID entity has the following attributes: parking ID number. The Time Taken entity has the following attributes: time taken in minutes. The Availability/Vacancy entity has the following attributes: availability/vacancy status. The Fine Generated entity has the following attributes: fine amount. The Spot Vacant entity has the following attributes: spot vacant status.

This ERD provides a clear and concise overview of the different entities in the parking lot database and the relationships between them.

* **Sequence Diagram:**



This is a sequence diagram that shows the steps involved in a car parking process. The diagram consists of three actors: Parking System, Database, and Car. The Parking System is the main actor, and it is responsible for initiating the parking process and interacting with the Database to store and retrieve parking information. The Database is responsible for storing and retrieving parking information, and the Car is the actor that is being parked.

The sequence diagram begins with the Parking System choosing a parking area and time. The Parking System then clicks the "Search" button, which initiates a transfer request to the Database. The Database then searches for a vacant spot in the chosen parking area and time. If a vacant spot is found, the Database returns the spot information to the Parking System. The Parking System then displays the available parking spot to the Car.

If the Car parks in the vacant spot, the Parking System requests the parking ID and time from the Car. The Parking System then generates a parking ID for the vacant spot and stores it in the Database. The Parking System then displays the parking information to the Car.

If the Car does not park in the vacant spot within 30 minutes, the Parking System generates a fine of Rs. 10 for each minute that the Car has exceeded the 30-minute time limit. The Parking System then displays the fine information to the Car.

1. **Miscellaneous** (Abbreviations, any other notations specifically used in the project, anything else that would help in better understanding by the review team and the client)

* **Abbreviation:**

**PMS:** Parking Management System

**ERD:** Entity-Relationship Diagram

**DB:** Database

**DS:** Data Structure

**CLI:** Command Line Input

* **Notation:**

**C language notation:** Data Structures of C language such as array have been used to store the unique parking ids, availability of a spot and fines.

**Command-Line Input/Output:** Command-line inputs may be denoted using angle brackets (e.g., <start> for the start location input).

* **Other Considerations:**

Determine how parking data will be stored and ensure data security and backup procedures.

Verify the system adheres to any relevant parking regulations or industry standards.

Design a user-friendly interface i.e., intuitive and easy to navigate for all user types.