

**SOFTWARE ENGINEERING AND PROJECT MANAGEMENT LAB**

***LAB FILE SUBMITTED BY: SUBMITTED TO:***

Dr. Arjun Arora

NAME: ANSHIKA SHARMA

SAP ID: 500107230

BATCH: B2 AIML NH

***Index***

|  |  |  |  |
| --- | --- | --- | --- |
| S. No. | EXPERIMENT | DATE | Remarks |
| 1. | Perform requirement analysis and to find the requirement specification of a given Problem | 5/02/2024 |  |
| 2. | To perform the function-oriented diagram using a structure chart: Flowchart | 12/02/2024 |  |
| 3. | To perform the user‘s view analysis for the given system - Use case diagram. | 26/02/2024 |  |
| 4. | To develop ER diagram to showcase structural view analysis. | 20/03/2024 |  |
| 5. | To develop Sequence diagram to showcase behavioral view analysis. | 20/03/2024 |  |
| 6. | To develop Software Requirement Specification (SRS) for taken up project |  |  |
| 7. |  |  |  |
| 8. |  |  |  |
| 9. |  |  |  |
| 10. |  |  |  |

**Experiment 1:** Perform requirement analysis and to find the requirement specification of a given Problem.

|  |  |
| --- | --- |
| **Project Title:** | **PARKING MANAGEMENT SYSTEM** |
| **Team Size:** | 4 MEMBERS |
| **Team Member Names:** | * ANSHIKA SHARMA * ANSHIKA AGARWAL * ANGELICA RAI * ANSH SAREEN |

**Requirements established after multiple brainstorming sessions:**

Requirement 1:-

Reduce the time consumption while parking

-By providing a readily available spot, we are trying to save the person’s time and would also prevent ruckus in the parking area.

Requirement 2:-

Python, VS Code

- We are using the platform VS Code and the programming language Python because of the diverse features of vs code and an easy to code language with availability of random number generation which we are using by importing library and availability of array data structures.

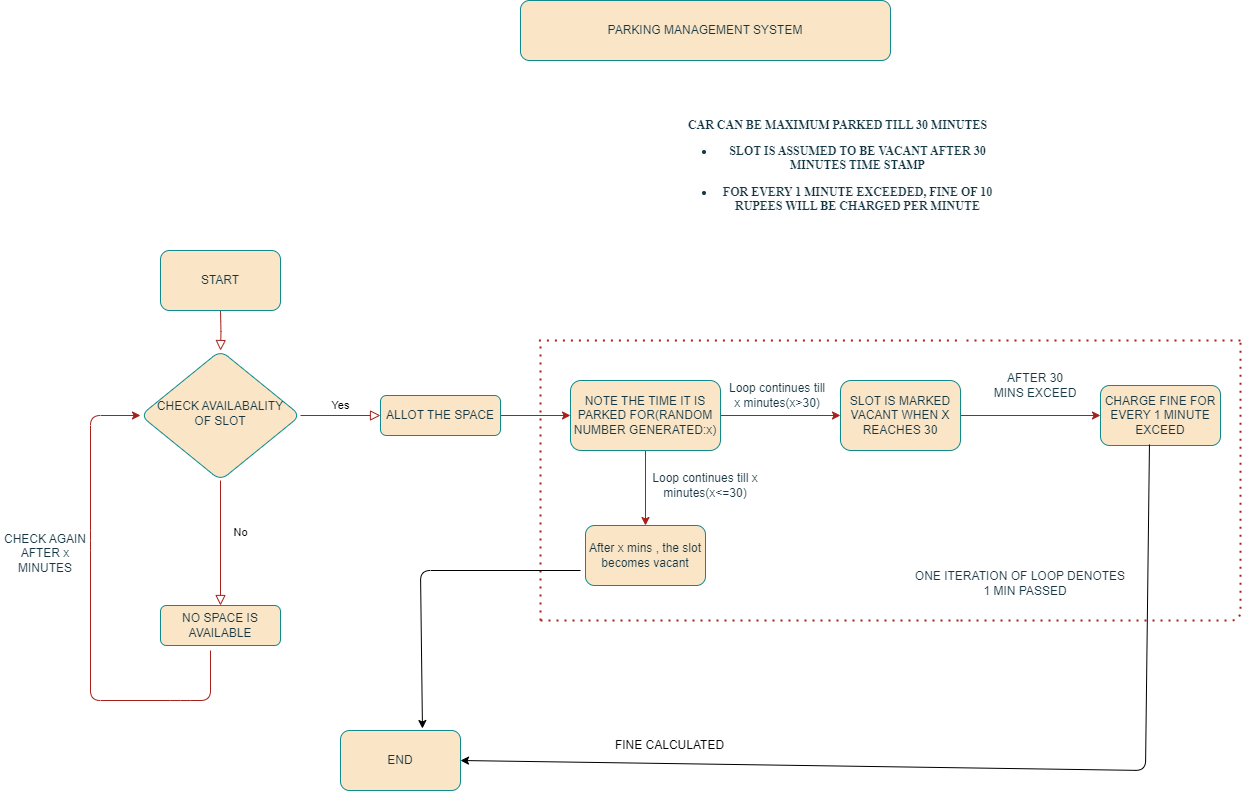
**Requirements mentioned above will be achieved by using the following:**

In its core, a parking management system aims to optimize the utilization of available parking spaces.

We are aiming at making a parking management system that will benefit us in reduced traffic congestion caused by circling for parking, improved air quality thanks to less time spent idling, and increased revenue for parking operators.

This is done by providing unique id to each slot and maintaining an array for the occupancy of the slots. We are also trying to maintain the smooth flow of operation of parking cars by setting a time limit which when exceeded will generate a fine for the person violating the time constraint. All of this will be implemented using python DS(array) and vs code will provide the right environment to execute the PMS.

**Experiment 2:** To perform the function-oriented diagram using a structure chart: Flowchart.

****

**Parking Management System: A Look at the Flowchart**

The flowchart above depicts a parking management system designed to automate the process of allocating parking spaces, tracking parking duration, and calculating parking fees. This system caters to short-term parking scenarios, with a maximum allotted time of 30 minutes. Let's delve into the system's functionalities and potential implications.

**Core Functionalities**

**Availability Tracking:** The system constantly monitors the availability of parking spaces. This real-time data is crucial for efficiently allocating spaces to arriving vehicles.

**Automated Allocation**: When a vehicle arrives, the system checks for available spaces. If a space is free, it's allocated to the vehicle, and the parking duration timer starts.

**Parking Duration Tracking:** The system meticulously tracks the time each vehicle is parked, forming the basis for calculating parking fees**.**

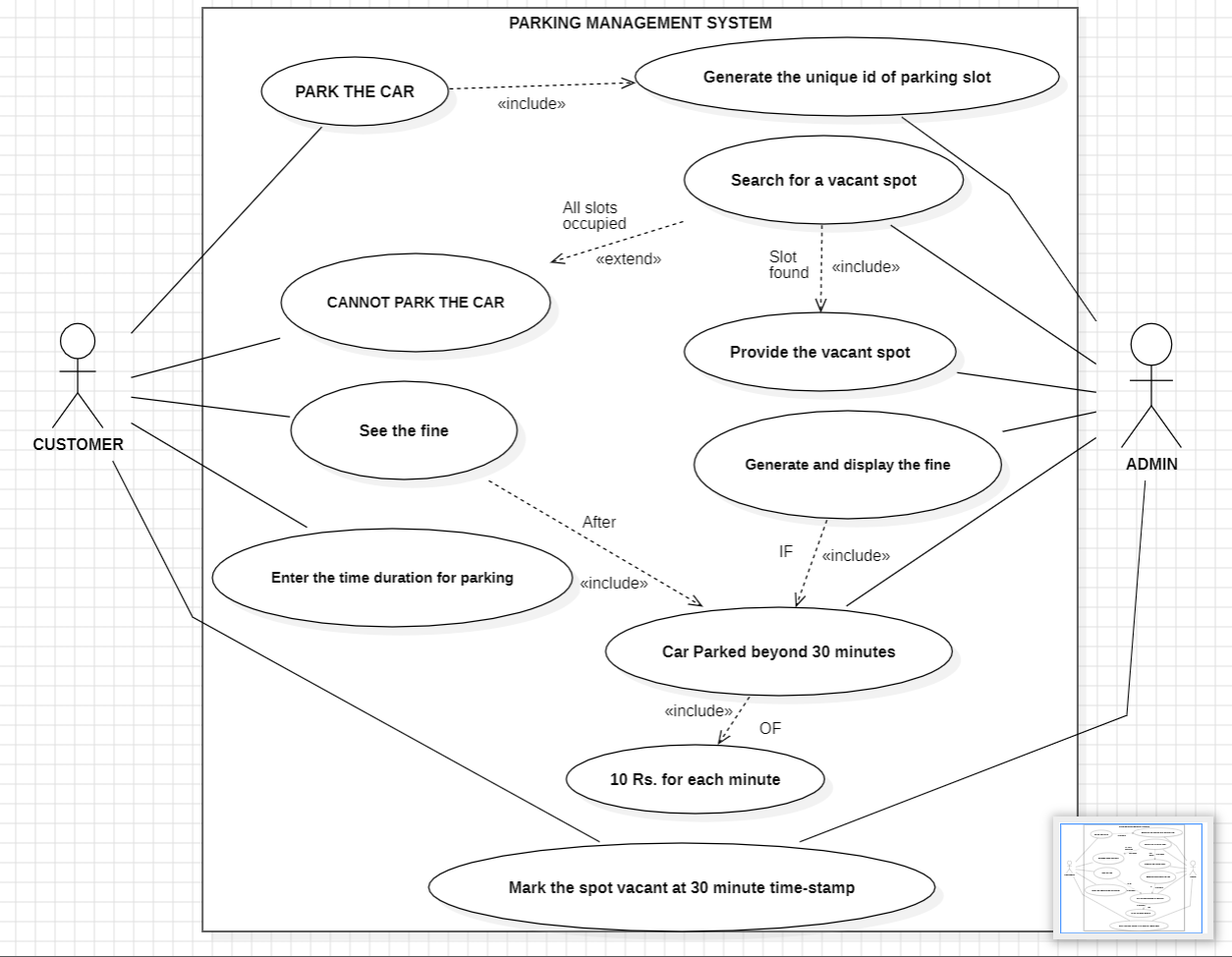
**Dynamic Fee Calculation:** The system employs a dynamic fee structure. Vehicles exceeding the 30-minute limit incur a fine of 10 rupees per minute of overtime. This incentivizes timely exits and discourages overstaying.

**Additional Considerations**

**Random Parking Duration:** The flowchart incorporates an interesting aspect: a random number generator determines the parking duration for each vehicle. This simulates real-world scenarios where parking durations vary depending on individual needs.

**30-Minute Limit:** The system enforces a strict 30-minute limit which may be the ideal limit for our model.

**EXPERIMENT – 3 :** To perform the user‘s view analysis for the given system - Use case diagram.



**EXPLANATION**

**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.

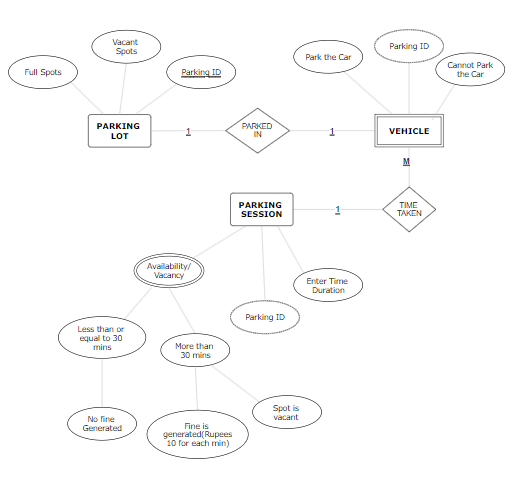
**See the Fine:** Allows the customer to view any fines associated with their parking.

**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.

**EXPERIMENT – 4 :** To develop ER diagram to showcase structural view analysis.



**EXPLANATION**

This is an **entity-relationship diagram (ERD)** for a parking lot database. It shows the different entities in the DB (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:**

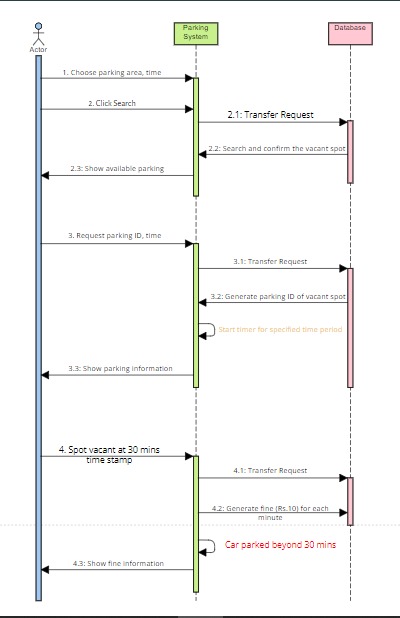
**The Parking Lot entity has a one-to-many relationship with the Parking Session entity** which means that each parking lot can have many parking sessions, but each parking session can only belong to one parking lot. **The Parking Session entity has a one-to-many relationship with the Vehicle entity** 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.

**EXPERIMENT – 5 :** To develop Sequence diagram to showcase behavioral view analysis.



**EXPLANATION**

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 DB to store and retrieve parking information. The DB 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.