# Documentation for Car Registration Search System

This document provides a comprehensive overview of the development and implementation of the **Car Registration Search System** using the C programming language. The system is designed to store and search car registration details (car number, owner, phone number, and location) from a predefined database. It provides an interactive console interface for users to search for specific car details.

1. Research

**Purpose:**

The goal is to create a system that stores and allows querying of car registration details. This search system can be beneficial for various real-world applications, such as in vehicle tracking, car rental services, parking management, or government databases.

**Background:**

Database Management: We are storing car details in a simple array of structures. A more complex version could integrate with a database management system (DBMS).

Search Algorithms: The system uses the linear search algorithm (i.e., iterating through the array) to locate a specific car registration number. The simplicity of the system allows for ease of use and quick development.

**Technology**:

C Programming Language: A procedural language suited for building console applications.

Standard Libraries:

`stdio.h` for input/output operations.

`string.h` for string manipulation functions like `strcmp`, `fgets`, and `strcspn`.

**Assumptions**:

- A limited dataset of car registrations (a maximum of 100).

- Car number input is consistent and properly formatted (e.g., no leading or trailing spaces).

**2. Analysis**

**Problem Analysis**:

Input: The system must accept a car number entered by the user, which will then be searched within an array of car registration structures.

Output: The system should output the details of the car (owner, phone number, and location) if the car number exists in the database. Otherwise, an appropriate message should be shown.

**System Requirements**:

Storage: The system will maintain an array of `Car Registration` structures, each holding the car number, owner's name, phone number, and location.

Search Functionality: We will implement a search function using a simple linear search. The function will iterate through the array, comparing the input car number with the stored car numbers.

3. Ideate

**System Design:**

Car Registration Structure:

`car Number [MAX\_LENGTH] `: To store the car number.

owner [MAX\_LENGTH] `: To store the owner's name.

phone Number [MAX\_LENGTH] `: To store the owner's phone number.

location [MAX\_LENGTH] `: To store the car's location.

Functions:

1.search Car (): A function that takes the car number as input and searches for it in the car registry array.

2. Main Function: The entry point of the program, where users are prompted to enter a car number to search.

Search Algorithm: Linear search is chosen for simplicity, given that the number of cars is limited.

User Interface:

A command-line interface (CLI) where the user can input a car number and receive feedback either with the car's details or a message stating the car is not in the database.

4. Build

Steps to Build:

1. Define the Structure:

Define a `Car Registration` structure to hold car details.

2. Create a Database (Array of Structures):

Define an array `cars []` that contains a list of car registrations.

3. Implement the `search Car` Function:

Loop through the array of cars, compare the input car number with each car's registration number, and display the matching record.

**4. Test**

Test Plan:

1. Test Case 1: Valid Car Number

Input: `MH 12 AG 1`

Expected Output:

Car Number: MH 12 AG 1

Owner: Aagam Gadiya

Phone Number: 9372024905

Location: Wagholi,Pune

2. Test Case 2: Car Number Not Found

Input: `MH 99 AB 1234`

Expected Output:

Car number MH 99 AB 1234 is not registered in the database.

3. Test Case 3: Invalid Input (Empty String)

Input: ` (Pressing Enter without entering anything)

Expected Output:

Error: Invalid input. Please enter a valid car number.

4. Test Case 4: Case Sensitivity Check

- Input: `mh 12 ag 1` (lowercase)

- Expected Output:

Car number mh 12 ag 1 is not registered in the database.

Edge Cases:

Car number with spaces at the beginning or end

Car number exceeding the maximum length (`MAX\_LENGTH`).

**5. Implementation**

Key Features:

The system is implemented using an array of structures.

It allows efficient searching within a small dataset (up to 100 cars).

Input validation ensures that the user enters a valid car number.

**6. Maintenance**

Future Improvements:

Dynamic Memory Allocation: Instead of using a static array, implement dynamic memory allocation to handle an arbitrary number of car registrations.

Advanced Search Algorithms: Implement binary search if the car registrations are sorted, improving the search efficiency.

Database Integration: Integrate with a real database system (e.g., SQLite, MySQL) to store and retrieve car details.

User Interface Enhancements: Build a graphical user interface (GUI) or a web-based interface for better user experience.