# A black and red text Description automatically generated

**PROJECT AND TEAM INFORMATION**

## Project Title

(Try to choose a catchy title. Max 20 words).

|  |
| --- |
| Parking Display System |

## Student/Team Information

|  |  |
| --- | --- |
| Team Name:  Team # (Mentor needs to assign) |  |
| Team member 1 (Team Lead) | Raj,Anjali-24011067  r[ajanjali4310@gmail.com](mailto:Rajanjali4310@gmail.com) |
| Team member 2 | Rawat,Aryan-24011067  [Aryan.rawat1508@gmail.com](mailto:Aryan.rawat1508@gmail.com) |
| Team member 3 | Bhasin,Manasvi-240222383  [Manasvibhasin107@gmail.com](mailto:Manasvibhasin107@gmail.com) |
| Team member 4 | Usman,Issac –240212281 |

**PROJECT PROGRESS DESCRIPTION (35 pts)**

## Project Abstract (2 pts)

(Brief restatement of your project’s main goal. Max 300 words).

|  |
| --- |
| The main goal of our project, **“Parking Management System,”** is to design and implement an efficient, user-friendly system that simplifies the process of vehicle parking using data structures in the C programming language. With the growing number of vehicles and limited parking spaces, manual management often leads to confusion, delays, and misuse of space. Our system aims to overcome these issues by providing a structured, automated approach for handling parking slots and vehicle records.  The project will simulate a parking area where vehicles are parked, removed, and managed according to availability. It will make use of **linked lists, queues, and stacks** to efficiently allocate and deallocate parking slots, ensuring smooth entry and exit operations. Each vehicle’s data, such as its registration number, arrival time, and slot number, will be stored for record-keeping. File handling will be used to maintain these records even after the program is closed, ensuring data persistence.  The system will feature a **menu-driven interface** where users can view available slots, park a vehicle, remove a vehicle, or check the waiting queue. Future integration possibilities include connecting the system with a graphical interface (Tkinter) or database (MongoDB) for a more interactive experience.  Ultimately, this project demonstrates how core data structure concepts can be applied to a real-life problem while enhancing management efficiency, accuracy, and user convenience. It also encourages modular programming and teamwork in developing practical C-based solutions . |

## Updated Project Approach and Architecture (2 pts) (Describe your current approach, including system design, communication protocols, libraries used, etc. Max 300 words).

|  |
| --- |
| Our project, **Parking Management System**, is being developed in the **C programming language** using a modular and data-driven approach. The system focuses on efficiently managing parking spaces, vehicle types, and cost calculation while ensuring clarity and convenience for users. The design integrates multiple functional modules such as *Slot Management*, *EV Charging Department*, *Billing System*, and *File Handling for Data Storage*.  The parking area is divided into **different floors based on vehicle type**:   * One floor is dedicated to **two-wheelers**, * Another for **four-wheelers**, and * A separate section for **electric vehicles (EVs)** equipped with charging points.   Each floor has a predefined number of slots, managed dynamically using **linked lists and queues**. A **linked list** structure maintains details of available and occupied slots, while a **queue** manages vehicles waiting for parking during full capacity. To enhance usability, the system includes **red and green indicators** to represent slot status — *green for available* and *red for occupied*.  The **EV Department module** allows users to charge their vehicles while parked, with additional cost automatically calculated based on the charging duration. The **Billing System** computes the total cost by considering parking time, vehicle category, and charging usage. Persistent data storage is achieved through **file handling**, ensuring that all vehicle and transaction records remain available after program termination.  Although no IoT or network protocols are currently used, the architecture is designed for future scalability — allowing integration with **database systems (MongoDB)** to provide enhanced visualization and real-time management.  Overall, the system demonstrates a structured, efficient, and scalable design that applies core **data structure concepts** to solve a real-world urban parking challenge. |

## 

## Tasks Completed (7 pts) (Describe the main tasks that have been assigned and already completed. Max 250 words).

|  |  |
| --- | --- |
| Task Completed | Team Member |
| 1. Available slot indicator 2. EV department with charging 3. Cost calculation | Aryan Rawat  Anjali Raj  Manasvi Bhasin |

## Challenges/Roadblocks (7 pts) (Describe the challenges that you have faced or are facing so far and how you plan to solve them. Max 300 words).

|  |
| --- |
| During the development of our **Parking Management System** in C, our team encountered several technical and design-related challenges. The main difficulties were associated with managing dynamic data, ensuring logical accuracy, and simulating real-world parking behavior within a console-based program.  One of the primary challenges was **implementing multiple floors for different vehicle types** (two-wheelers, four-wheelers, and electric vehicles). Designing an efficient data structure to handle each floor separately without redundancy required careful use of **linked lists and queues**. To address this, we modularized the program so that each floor operates as an independent list, enabling clearer management of slots and vehicles.  Another significant challenge was **real-time slot indication** using red and green indicators within a text-based interface. Since the C language does not inherently support graphical visualization, representing color indicators accurately required experimenting with terminal color codes and formatted output to simulate a visual effect.  We also faced issues with **cost calculation logic**, especially when combining different billing components such as duration, vehicle type, and EV charging fees. To resolve this, we created a structured algorithm that separates parking cost and charging cost calculations for better accuracy.  Additionally, ensuring **data persistence** through file handling presented initial complications, such as incorrect overwriting or improper data retrieval after program termination. This was resolved by implementing proper file opening modes and structured data storage using formatted file I/O.  Future challenges include **integrating a MongoDB database** for enhanced interaction and scalability. These will be addressed by gradually shifting from a pure console application to a modular system that supports external connectivity while preserving core C logic. |

## Tasks Pending (7 pts) (Describe the main tasks that you still need to complete. Max 250 words).

|  |  |
| --- | --- |
| Task Pending | Team Member (to complete the task) |
| Integration of the project , implementation  Testing and validation | Manasvi Bhasin |

## 

## Project Outcome/Deliverables (2 pts)

(Describe what are the key outcomes / deliverables of the project. Max 200 words).

|  |
| --- |
| The key outcome of the **Parking Management System** project is the development of an efficient and user-friendly application that automates the parking process using fundamental data structures in the **C programming language**. The system will successfully demonstrate how linked lists and queues can be applied to real-world problems involving dynamic allocation and resource management.  The major deliverables include:   * A **functional console-based parking system** capable of managing multiple floors for two-wheelers, four-wheelers, and electric vehicles. * A **slot availability feature** displaying occupied and vacant slots using red and green indicators. * An integrated **EV department module** that manages charging stations and calculates additional charging costs. * A **cost calculation module** that automatically computes total parking fees based on vehicle type, parking duration, and charging usage. * **File handling implementation** for maintaining parking records and transaction history persistently.   The project will deliver a working model that improves parking space utilization, minimizes human error, and simplifies management tasks. It will also serve as a foundation for future integration with  **databases (MongoDB)** to make the system more interactive, scalable, and suitable for real-time use. |

# Progress Overview (2 pts) (Summarize how much of the project is done, what's behind schedule, what's ahead of schedule. Max 200 words.)

|  |
| --- |
| The **Parking Management System** project is currently in its mid-development phase, with several major modules already completed and successfully tested. The modules for **slot availability**, **EV department management**, and **cost calculation** have been fully implemented. The slot availability system uses red and green indicators to represent occupied and free parking spaces, while the EV department efficiently manages charging points and additional billing for electric vehicles. The cost computation module accurately calculates parking fees based on time, vehicle type, and charging usage.  In terms of progress, the project is **on schedule** with its core functionality and logic implementation. The **file handling** module for data storage and record retrieval has also been integrated and tested successfully. The system currently runs smoothly as a **console-based application**.  Tasks that are **behind schedule** include the development of a **graphical interface (Tkinter)** and integration with a **database (MongoDB)**, which were planned as enhancement features. However, these are secondary goals and do not affect the primary functionality of the project.  Overall, the project demonstrates steady progress, with approximately **80–85% completion**, and the team is confident about finalizing testing and minor improvements before the final evaluation. |

# Codebase Information (2 pts) (Repository link, branch, and information about important commits.)

|  |
| --- |
| Your answer here |

## Testing and Validation Status (2 pts) (Provide information about any tests conducted)

|  |  |  |
| --- | --- | --- |
| Test Type | Status (Pass/Fail) | Notes |
|  |  |  |

# Deliverables Progress (2 pts) (Summarize the current status of all key project deliverables mentioned earlier. Indicate whether each deliverable is completed, in progress, or pending.)

|  |
| --- |
| Your answer here |