DESIGN DOCUMENT - ChargeMate D7

Team no.: **37**
Members:

2021111013 Bhav Beri
2021114009 Aditya Raghuvanshi
2021101098 Himanshu Sarraf
2021113012 Aryan Gupta

Introduction

1) The purpose of a design doc for this project is to provide a clear and comprehensive overview of the app's features, requirements, technical details, and how it interacts with the autonomous robot for car charging.
2) The design doc should clearly state the project's goals, such as developing a user-friendly app that enables car owners to request and monitor charging using an autonomous robot.

System Overview

Software system

- Mobile and Web App: The mobile and web app provides an interface for users to interact with the system. The app is responsible for user authentication, charging requests, payment processing, and user notifications about charging status. The mobile app should be designed to be user-friendly and accessible to all users.
- Mobile app: It's a complete app based on User-interface where the user can call the bot and charge his/her car and can monitor the car charging using his phone, and can even make payments via an in-app wallet.
- Web app: The web app is completely based for the use of admin through which he can add new parking places, upload the map of parking places, and monitor the existing parking places and robot.

Real-life Scenarios

- Parking garages: In densely populated areas, parking garages are common. The app could be used to navigate the autonomous robot to the parking spots where cars are parked.
- Remote areas: In remote areas with no charging stations, the app could be used to deploy the autonomous robot to charge electric vehicles.
- Shopping centers: Shopping centers are another location where an autonomous robot for car charging app could be useful.

Problems solved

Parking and charging management: The app can help users to locate available charging stations and reserve a spot for their vehicle. The admin panel can help to park and charging operators manage the charging infrastructure, monitor usage, and optimize charging schedules.

Payment and billing: The app can provide users with a seamless payment and billing process, including options for paying by wallet.

User experience: The app can provide users with a seamless and intuitive interface that simplifies finding and using charging stations. The admin panel is a robust system that handles all the administration parts.

Sustainability: EV charging is essential to the transition to a sustainable transportation system. The app can help promote the use of electric vehicles by making it easier for drivers to find charging stations and reducing the need for fossil fuels.

Design Overview

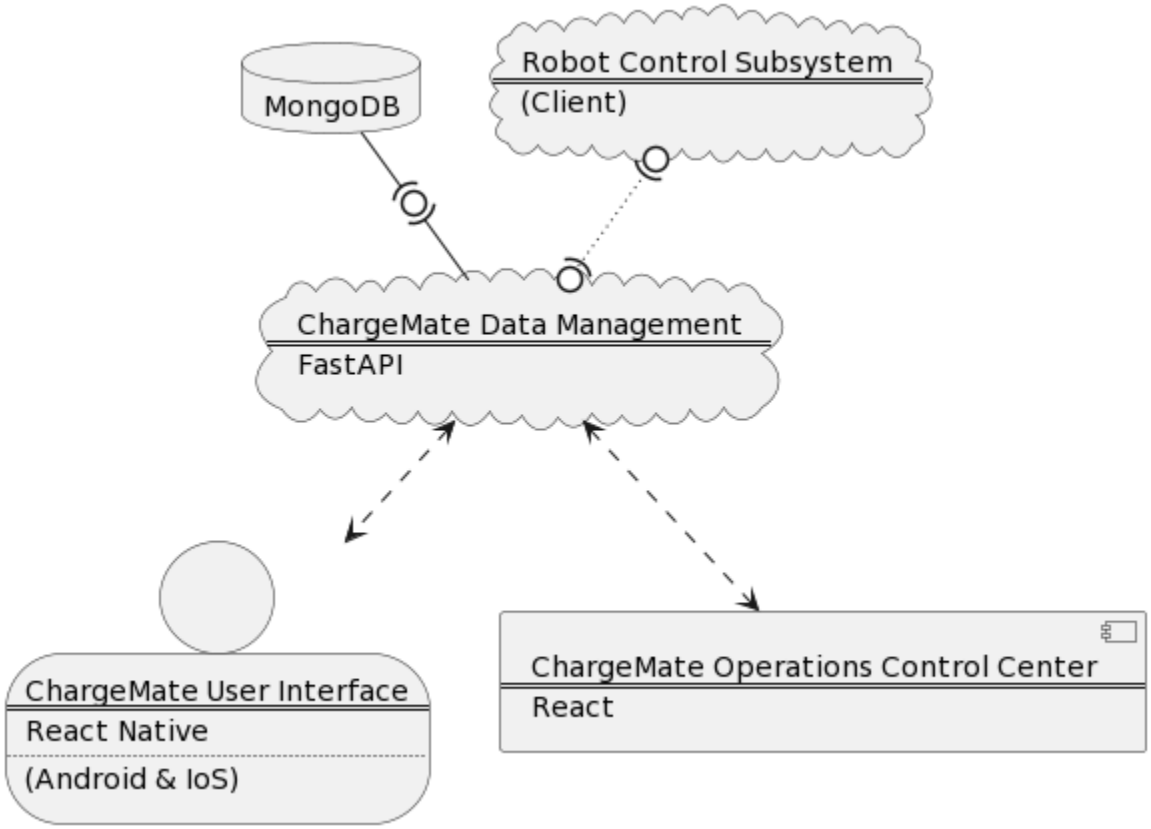
Architectural design

1) **Robot Control Subsystem** (ROS): This subsystem is responsible for coordinating the overall operation of the robot. It receives input from the Navigation and Charging subsystems and uses this information to determine the robot's next actions. The Control Subsystem also monitors the state of the charging station and the cars parked there and decides when and how to charge each car.
2) **ChargeMate User Interface**: This module is designed to provide a seamless and user-friendly experience for electric vehicle (EV) owners seeking to charge their vehicles. The module is built as a mobile application, which will serve as the primary interface for users to interact with the charging system. Starting from booking a charging slot for the bot, users will be able to monitor the charging process in real time and receive updates on the status of their vehicle. The application will also provide an easy-to-use platform for users to carry out financial transactions, ensuring a smooth and streamlined experience.
3) **ChargeMate Operations Control Center** (Web Application): The ChargeMate Operations Control Center is a web-based portal for company administrators and robot managers. Its purpose is to provide a centralized platform for these individuals to manage and monitor the charging bots, ensuring smooth and efficient system operations. The portal offers real-time statistics and status updates, allowing administrators to keep track of the charging process and quickly identify and resolve any issues that may arise. The platform also provides a comprehensive suite of tools and features, enabling administrators to modify and adjust the charging bots as and when needed. The ChargeMate Operations Control Center has a user-friendly interface and

advanced security measures, ensuring only authorized personnel can access and manipulate the system.

4) **ChargeMate Data Management** (Backend): This module forms the backbone of the ChargeMate platform and is responsible for managing and storing all the data generated by the system. It acts as the server layer that connects the web-based portal for administrators and the mobile application for users, facilitating communication and data transfer between the two. The ChargeMate Data Management Module is designed to interact with a robust and secure database to store information on charging sessions, financial transactions, and other relevant data. This information will generate real-time statistics and analytics, which administrators can access through the web-based portal. In addition to data storage and management, the module is responsible for handling complex algorithms and processes to ensure a smooth and efficient operation of the ChargeMate system.

We can find a small relationship between the above subsystems in the diagram below:



Link - www.plantuml.com/plantuml/png/NP11JuD048NlyoIcPE1GWdSrhJGc753JnbFZOO0HIM2csHrClFIt2oufRUUmEzdaU-_jMtiZPqFVGKr69Nd6ytA0GgNHxJDy0bIT3ZM6ym3PDxc62pB6BHbXGK8Dzom6Z-F02tdRlESJU6JdLJ1Sr-Y7PuUv6BilgZYGEwRAy9Mi_M58ne5e8xNJjiOxp7KV_uVH_gZ2OhXiLyflenrPg-8nKp6d7MOy6LsOtFpgVbJOQQbs9lP3wN-zSH-WAEjQ5fi2B6MjeZHU1O5Px259dfPObu-X0fZTqlXqYj8PNRFKGz_z0G00

System Interface

User interface

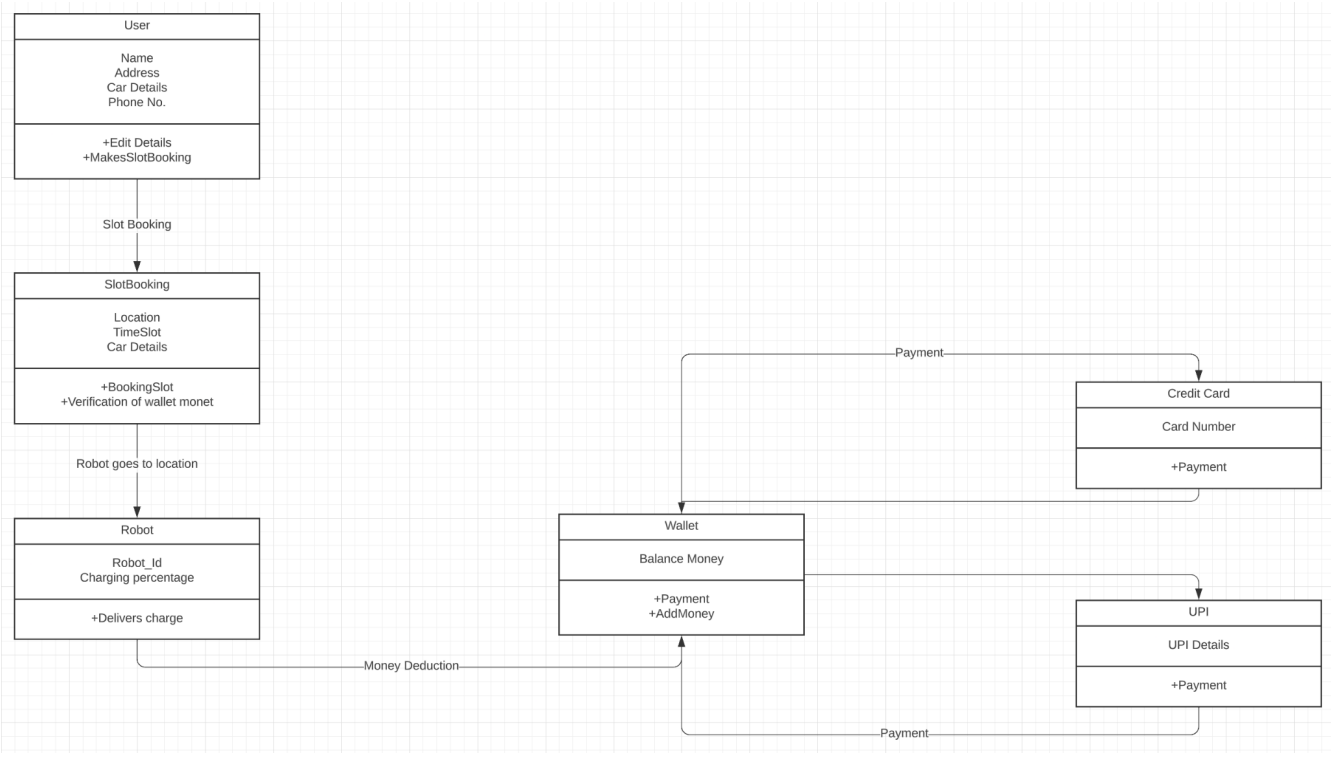
Users could log in to the application using their credentials (If new users, they may register as new users). Next, the user could see his/her profile easily and make edits to it if needed. On the main screen, the user could see options to select his location (parking location) and start the charging slot booking process. The user can then select the options on the screen for the charging bot. Once everything is selected, the charging process will start (the bot will come, the user will connect the charger, and so on), and then the user will be able to see live updates on the screen for the charging process. Once done, the financial process from the user's wallet would happen. After it, the user could get the statistics for the last charge.

API's

Our frontend components make REST API requests to the backend routes, but our product does not have any API which a user can call independently in his own program.

Model

UML diagram



Link for UML diagram -
https://lucid.app/lucidchart/92743122-06bd-4230-85ac-96411823418a/edit?viewport_loc=29%2C17%2C2209%2C1298%2CHWEp-vi-RSFO&invitationId=inv_96a5956d-b31d-494b-a6b5-1f6f453d25d6

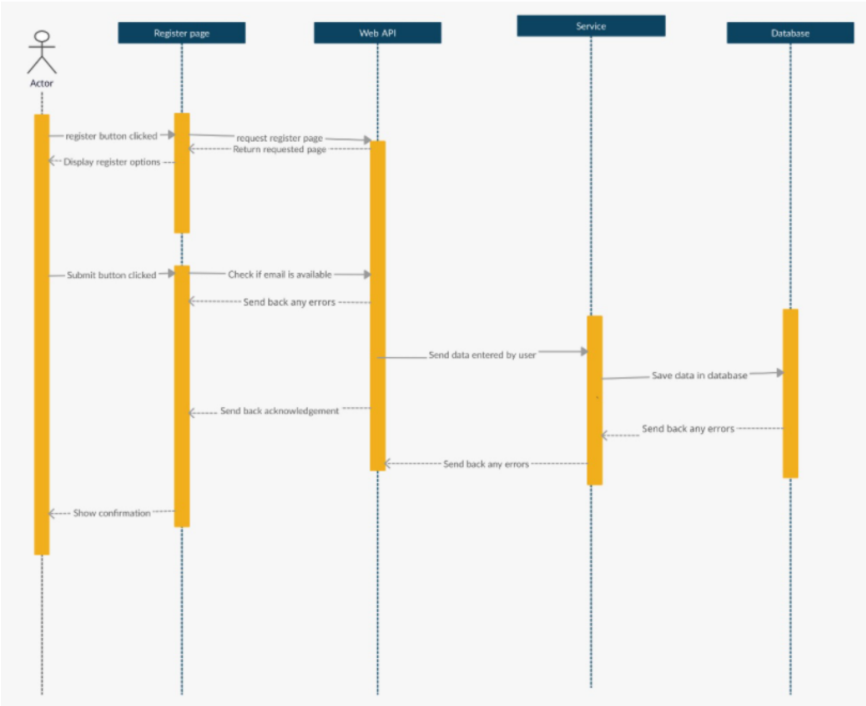
Classes Description:

<table>
<tr><td rowspan="2">User</td><td>Class state: The class is responsible for keeping track of the following info on the user.
Username
Email
Contact
Full_name(optional)
password</td></tr>
<tr><td>Class behavior: This class has the following methods
Update Profile(): used to update the profile details
Sign_Out(): The user can log out
create_user(): creates User
Sign_in(): The user can log in
get_user(): Fetches the details of the user.</td></tr>
<tr><td rowspan="2">Wallet</td><td>Class state: The class is responsible for all the monetary-related stuff.
USER_ID
Balance</td></tr>
<tr><td>Class behavior: This class has the following methods
Add_money() - used to add money to the wallet.
Get_status () - fetches the status of the balance of the user.
Deduct () - deducts the money from the wallet after the use of the service.</td></tr>
</table>

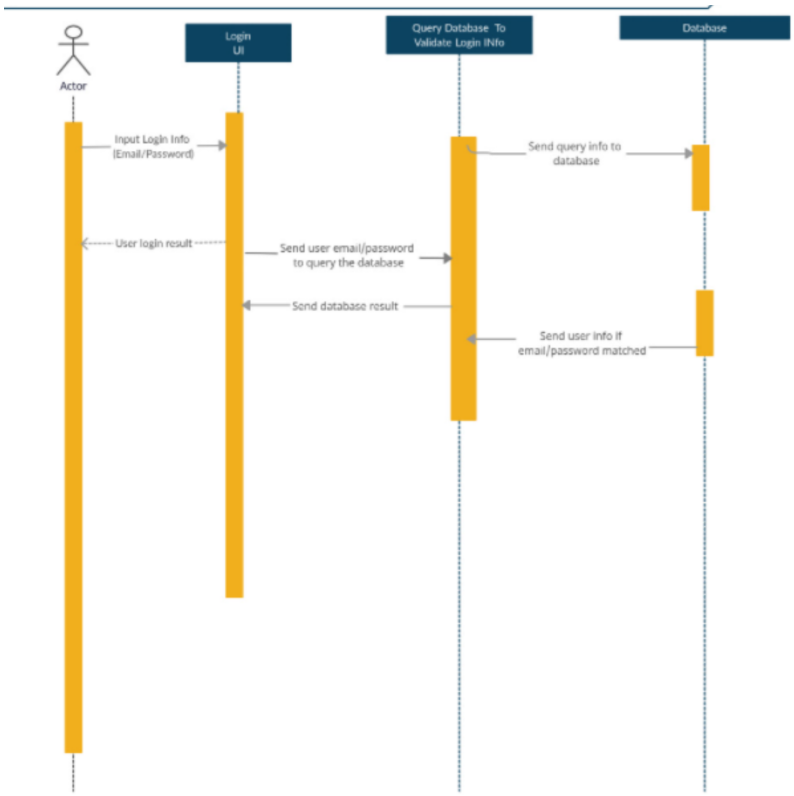
Robot	Class state: The class is responsible for robot-related works • Robot_id • Charging_percentage :
	Class behavior: This class has the following methods • Add_Robot () - Used to add new robots • Get_status () - gives us the status for its battery or location. • Destination () - Gives us the destination it is heading towards when called.
Admin	Class state: The class is responsible for keeping track of the following info on the Admin. • Admin_id • Password • number
	Class behavior: This class has the following methods • New_Location () - Used to add new locations • Login () - Logs in to the admin • Logout () - Logs out the admin • Get_Details - fetches the details of robots present in their location.
Transaction	Class state: The class is responsible for keeping track of the transactions. • Current • History
	Class behavior: This class has the following methods • New () - makes the new transactions • History () - fetches the history of all transactions.

Sequence diagram (UML)

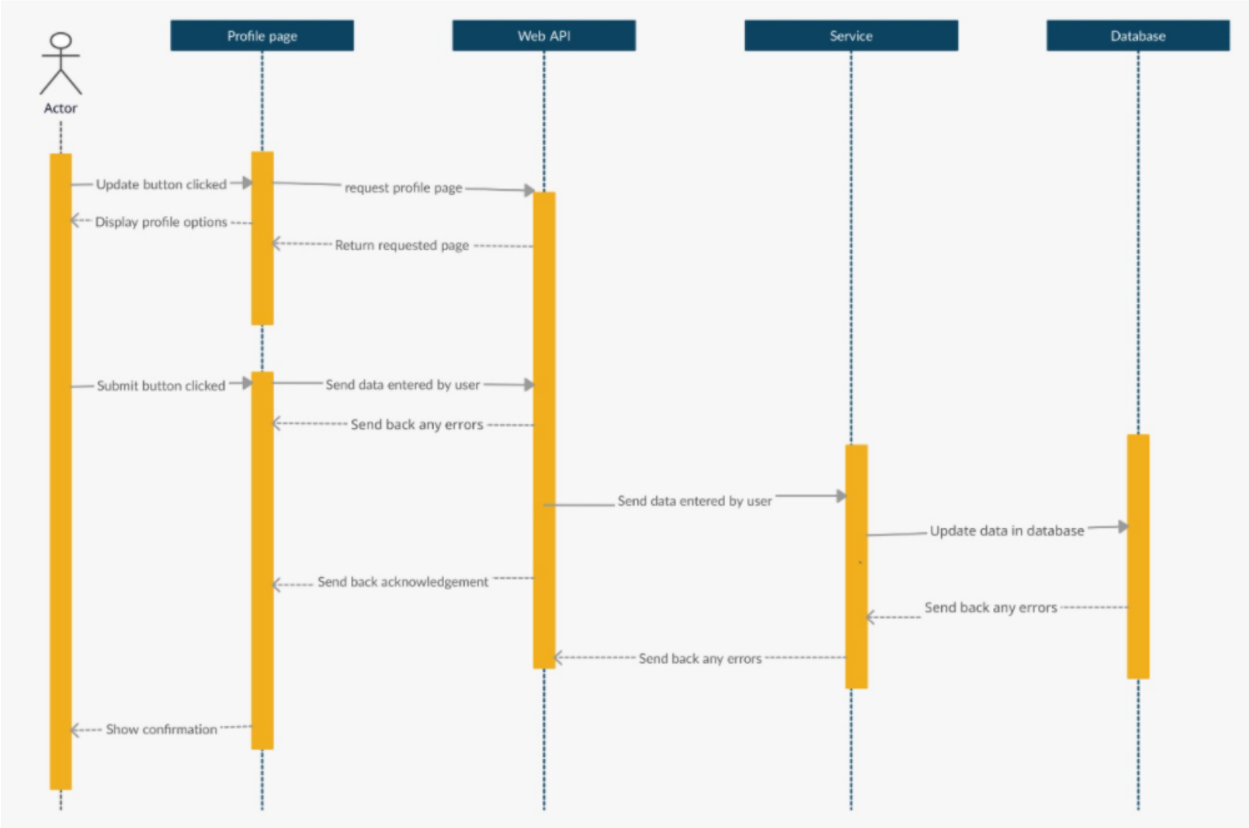
User Registration



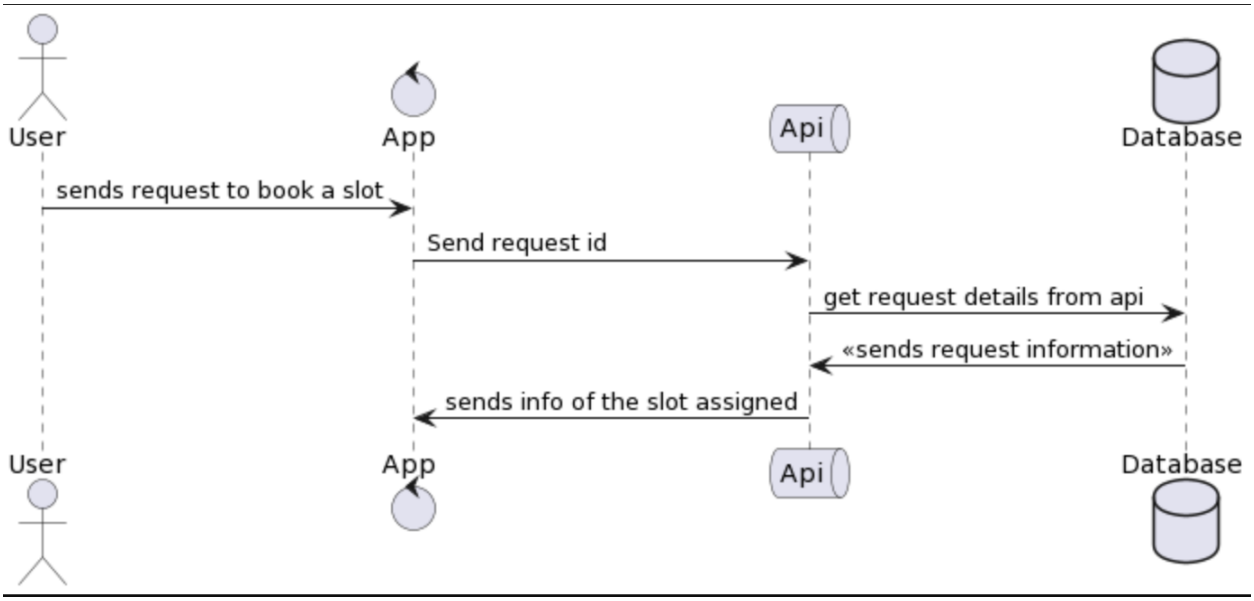
Login



Update Profile



Booking slots



Design Rationale & Issues

The running list of issues that may arise during the project:

1) Navigation and obstacle avoidance:
   The robot must be able to navigate to the car and avoid any obstacles in its path.
2) Communication:
   The robot must be able to communicate with the car and the charging station to determine when and where to charge the car.
3) Security:
   As the robot will be interacting with sensitive electronic equipment, it must be designed to prevent unauthorized access and ensure the security of the charging process.
4) Scalability:
   As the number of electric vehicles on the road continues to grow, the demand for autonomous robots for car charging is likely to increase. Ensuring the scalability of the technology will be an important consideration for developers.

Alternatives that can be designed in the future for the issues mentioned above:

1) One possible alternative to the issue of navigation in autonomous robots for car charging is to implement a centralized control system. This would involve a central computer or server that would be responsible for coordinating and directing the actions of the autonomous robots.
2) Another alternative solution could be to use visual markers or beacons placed near the charging stations that the robot can detect using computer vision techniques. These markers could provide the robot with information about the location and orientation of the charging station, allowing it to accurately dock with the station without the need for communication.
3) Implementing secure communication protocols: Another approach is to use secure communication protocols such as HTTPS or SSL/TLS to encrypt the data transmitted between the robot and the charging station. This helps to prevent any unauthorized access or interception of data. Implementing multi-factor authentication: One way to enhance security is to require multiple forms of authentication before the autonomous robot is allowed to initiate the car charging process.
4) There are several possible alternatives for addressing the scalability issues:
   - Cloud Computing: Another alternative is to use cloud computing services to offload some of the computational load from the robots to remote servers.

- Distributed Task Management: A third option is to use distributed task management systems that allow multiple robots to work together to complete a task, such as charging a vehicle.
- Hybrid Approaches: Finally, a combination of these approaches may be the most effective solution, depending on the specific requirements and constraints of the system.