

PROFESSIONAL TRAINING REPORT - II

entitled

ONWHEELS:CENTRALIZED MANAGEMENT SYSTEM

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering with
specialization in Internet of Things

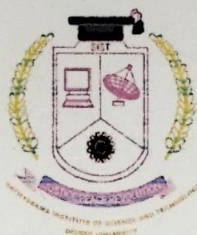
by

D.Bala Chandu [Reg.No.41732004]

R.Durga Naga Prasad [Reg.No.41732021]

M.Karthik [Reg.No.41732028]

K.Jithendar [Reg.No.41732027]



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

**Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119**

MAY 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

[DEEMED TO BE UNIVERSITY]

CATEGORY-I UNIVERSITY BY UGC

Accredited "A++" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Professional Training Report is the bonafide work of **Mr.D.Bala Chandu (41732004)** who carried out the project entitled **"OnWheels:centralized management system"** under my supervision from January 2024 to May 2024.

Internal Guide
Mrs. K. ABINAYA M.E.,(Ph.D)

Head of the Department
Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on 06-05-24

Internal Examiner

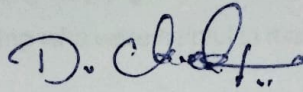
External Examiner

DECLARATION

I, D.Bala Chandu (Reg.No-41732004), hereby declare that the Professional Training Report-II entitled "OnWheels:centralized management system" done by me under the guidance of Mrs. K. ABINAYA M.E,(Ph.D), is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering with specialization in Internet of Things.

DATE: 6-05-2024.

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. S.Vigneshwari M.E., Ph.D., Head of the Department of Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Internal Guide **Mrs.K.ABINAYA M.E,(Ph.D)** for her valuable guidance, suggestions and constant encouragement which paved way for the successful completion of my phase-2 Professional Training.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

COURSE CERTIFICATE

WELCOME - HCL-Sathyabama - Mini-Project Internship Inbox x



Sankaran V <sankaran.v@hcl.com>

to Sankaran ▼

Good Evening HCL-Sathyabama Mini-Project Interns,

I have BCCed all Interns and Director Placement, Professors, Deans and HODs.

This is Dr.V.Sankaran, HCL Technologies.

Happy to WELCOME you all to the mini-project internship.

ABSTRACT

The OnWheels:centralized management system aims to revolutionize the management and maintenance of vehicles through the implementation of a comprehensive software solution. This application, developed with a focus on user-friendly design and efficiency, offers a wide range of functionalities tailored to meet the diverse needs of vehicle owners, service providers, and fleet managers. At its core, the application serves as a centralized platform for managing various aspects of vehicle servicing, including scheduling appointments, tracking maintenance history, live GPS tracking functionality, and facilitating communication between vehicle owners and service centers. Built upon robust PHP technology, the application provides a scalable and reliable framework for seamless integration with existing systems and future expansions. Its intuitive interface ensures ease of use for both administrators and end-users, facilitating smooth navigation and efficient task completion. Moreover, the application leverages advanced features such as automated reminders and notifications to enhance user engagement and improve service delivery. One of the key highlights of the Vehicle Service Application is its ability to streamline operations and enhance customer satisfaction through real-time tracking of service requests and vehicle statuses. By providing transparency and visibility throughout the servicing process, the application enables vehicle owners to stay informed and empowered, while service providers can optimize resource allocation and With real-time GPS tracking, businesses can track their vehicles' locations at any given moment, providing valuable insights into fleet operations and enabling timely decision-making. The inclusion of geofencing functionality further enhances monitoring capabilities by allowing users to define virtual boundaries and receive alerts for entry and exit events within specified areas. This proactive approach to monitoring helps businesses ensure compliance with route schedules, enhance security, and improve overall operational efficiency.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	vii
1	INTRODUCTION	1
	1.1 Overview	
2	LITERATURE SURVEY	2
	2.1 survey	
3	REQUIREMENTS ANALYSIS	
	3.1 Objective	4
	3.2.1 Hardware Requirements	
	3.2.2 Software Requirements	5
4	DESIGN DESCRIPTION OF PROPOSED PRODUCT	9
	4.1.1 Ideation Map/Architecture Diagram	9
	4.1.2 Various stages	10
	4.1.3 Internal or Component design structure	12
	4.1.4 working principles	14
	4.2 4.2.1 Novelty of the Project	17
5	RESULTS AND DISCUSSION	19
6	SUMMARY AND CONCLUSIONS	24
	REFERENCES	25
	APPENDIX	
	A. Research Paper	26
	B. Source Code	34

LIST OF FIGURES

Figure No.	Figure Name	Page No.
Fig 4.1	Ideation Map/System Architecture	10
Fig 4.2	Working Principle Architecture	16
Fig 5.1	Fronted Booking	19
Fig 5.2	Dashboard	19
Fig 5.3	Lists of Vehicles	20
Fig 5.4	Vehicle Details	20
Fig 5.5	Add Vehicle	21
Fig 5.6	Lists of Drivers	21
Fig 5.7	Lists of Bookings	22
Fig 5.8	Lists of Customers	22
Fig 5.9	Lists of Fuels	23
Fig 5.10	Income and Expenses	23

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The OnWheels:centralized management system project represents a cutting-edge solution in fleet management, offering an array of functionalities tailored to meet the dynamic needs of vehicle owners, service providers, and fleet managers. Leveraging advanced features such as live GPS tracking and geofencing, this application sets a new standard in vehicle management and servicing.

Built upon a robust PHP framework, the Vehicle Service Application empowers users to efficiently manage vehicles, drivers, customers, income, expenses, and detailed reports. Real-time GPS tracking functionality ensures heightened visibility into vehicle locations, allowing for optimal fleet management and enhanced security. Moreover, the inclusion of geofencing capabilities enables precise monitoring of inbound and outbound activity within designated areas, bolstering operational efficiency and compliance.

A standout feature of the OnWheels:centralized management system is its provision of unique tracking URLs for trips, allowing customers to seamlessly track the progress of their vehicles in real-time. This transparent approach to service tracking fosters trust and satisfaction among users, enhancing the overall service experience.

As the automotive industry continues to evolve, the Vehicle Service Application emerges as a pivotal tool for modernizing vehicle management and servicing operations. By offering a comprehensive suite of features and leveraging cutting-edge technologies, this application is poised to streamline processes, optimize resource utilization, and drive customer satisfaction to new heights.

As businesses increasingly recognize the importance of efficient fleet management, the Fleet Management System with Live GPS Tracking emerges as a game-changer. By providing a comprehensive suite of tools for vehicle tracking, expense management, and customer communication, it sets a new standard for operational excellence in the automotive industry.

CHAPTER 2

LITERATURE REVIEW

2.1 SURVEY

Vehicle Routing Problem (VRP) is a classic optimization challenge in logistics and transportation management. This paper provides a comprehensive review of VRP variants, solution methods, and recent advances, focusing on their application in fleet management systems for efficient vehicle service scheduling and routing. [1]

Real-Time GPS Tracking Systems have become essential tools for fleet management and vehicle service applications. This survey paper explores the evolution of GPS technology, its integration with vehicle service applications, and its impact on improving operational efficiency, customer satisfaction, and safety. [2]

Machine Learning Applications in Predictive Maintenance have gained significant attention in recent years for their potential to optimize vehicle service operations. This review paper discusses various machine learning algorithms, data sources, and case studies relevant to predictive maintenance in automotive fleets. [3]

Customer Relationship Management (CRM) Systems play a crucial role in vehicle service applications by facilitating communication and enhancing customer satisfaction. This literature survey examines CRM strategies, technologies, and best practices for implementing effective customer relationship management in the automotive industry. [4]

Geofencing Technology has emerged as a valuable tool for monitoring vehicle movements and enhancing security in fleet management systems. This survey paper explores the principles, applications, and benefits of geofencing in vehicle service applications, with a focus on real-world implementation challenges and solutions. [5]

Cloud Computing in Fleet Management Systems offers scalability, flexibility, and cost-effectiveness for hosting vehicle service applications. This literature review discusses cloud computing concepts, service models, and deployment strategies relevant to fleet management, highlighting the advantages and challenges of adopting cloud-based solutions. [6]

Mobile Applications for Vehicle Service Booking have become increasingly popular among consumers seeking convenient and efficient ways to schedule service appointments. This survey paper examines the features, design considerations, and user adoption factors of mobile apps in the context of vehicle service applications. [7]

Blockchain Technology holds promise for enhancing trust, transparency, and security in vehicle service applications. This review explores the potential applications of blockchain in automotive service management, including supply chain traceability, vehicle history tracking, and secure payment processing. [8]

Internet of Things (IoT) Devices are revolutionizing vehicle service applications by enabling real-time monitoring, predictive maintenance, and data-driven decision-making. This literature survey discusses IoT principles, applications, and challenges in the context of fleet management and vehicle servicing. [9]

Human-Computer Interaction (HCI) Principles are essential for designing user-friendly interfaces in vehicle service applications. This paper reviews HCI concepts, methodologies, and design guidelines relevant to the development of intuitive and accessible interfaces for automotive service management systems. [10]

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 OBJECTIVE OF THE PROJECT

The primary objective of the OnWheels:centralized management system project is to develop a comprehensive software solution that revolutionizes the management and maintenance of vehicles, catering to the needs of vehicle owners, service providers, and fleet managers. This project aims to address the existing challenges and limitations in traditional vehicle servicing methods by leveraging technology to enhance efficiency, transparency, and user experience. The following are the specific objectives of the project:

1.Streamlined Service Management: The project aims to streamline the process of scheduling, tracking, and managing vehicle service appointments. By providing a user-friendly platform accessible via web and mobile devices, the application will enable vehicle owners to easily book service appointments, select preferred dates and times, and receive real-time updates on service progress.

2.Enhanced Communication: Effective communication is essential for ensuring a seamless vehicle servicing experience. The project seeks to facilitate communication between vehicle owners and service centers by providing features such as appointment reminders, service status notifications, and feedback mechanisms. By keeping users informed and engaged throughout the servicing process, the application aims to improve customer satisfaction and loyalty.

3.Real-Time Tracking and Monitoring: One of the key objectives of the project is to incorporate real-time GPS tracking and monitoring functionality into the application. This feature will enable vehicle owners and service providers to track the location and status of vehicles in real-time, ensuring timely service delivery, optimizing resource allocation, and enhancing security.

4.Comprehensive Service History: The project aims to provide vehicle owners with access to a comprehensive service history for their vehicles. By maintaining detailed records of past service appointments, maintenance tasks, and repairs, the application

will empower users to make informed decisions regarding future servicing needs and vehicle maintenance.

5.Integration of Payment and Invoicing: To streamline the payment process, the project will integrate secure payment gateways into the application, allowing users to make payments for services directly through the platform. Additionally, the application will generate digital invoices for transparency and record-keeping purposes, simplifying the billing process for both vehicle owners and service providers.

6.Analytics and Reporting: The project aims to incorporate analytics and reporting tools into the application to provide insights into service trends, customer behavior, and business performance metrics. By analyzing data collected from service appointments, user interactions, and feedback, the application will enable service providers to make data-driven decisions, identify areas for improvement, and optimize service delivery.

The OnWheels:centralized management system project seeks to develop a comprehensive software solution that enhances the vehicle servicing experience for all stakeholders involved. By focusing on streamlining service management, improving communication, incorporating real-time tracking and monitoring, maintaining comprehensive service history records, integrating payment and invoicing functionalities, and providing analytics and reporting capabilities, the project aims to deliver a user-friendly, efficient, and transparent platform that sets new standards for vehicle servicing in the automotive industry.

3.2 REQUIREMENTS

3.2.1 *HARDWARE REQUIREMENTS*

1.Server Infrastructure:

- High-performance servers capable of running PHP applications efficiently.
- Adequate CPU resources to handle PHP execution and application logic processing.
- Sufficient RAM to accommodate PHP runtime memory requirements and ensure smooth application performance.
- Ample storage space to host the application codebase, database files, and other system files.

- SSD storage drives or equivalent for faster read/write operations and improved overall system responsiveness.

2.Database Server:

- Dedicated database server compatible with PHP and capable of supporting the required PHP extensions (e.g., PDO).
- Support for the chosen database management system (e.g., MySQL, PostgreSQL) with appropriate version compatibility.
- Adequate CPU and RAM resources to handle database queries, indexing, and data manipulation efficiently.
- Redundant storage solutions or RAID configurations to ensure data integrity and minimize the risk of data loss.

3.Networking Equipment:

- Network switches, routers, and other networking equipment to facilitate communication between server components and client devices.
- Gigabit Ethernet or higher-speed connections to ensure low latency and high throughput for data transmission.
- Firewall and network security appliances to protect against unauthorized access, malware, and other cyber threats.

4.Backup and Disaster Recovery Systems:

- Backup servers or storage devices for regular data backups, including application files, databases, and system configurations.
- Automated backup solutions with scheduling capabilities to ensure regular backups without manual intervention.
- Disaster recovery plans and protocols to minimize downtime and data loss in the event of hardware failures, natural disasters, or other emergencies.

5.Monitoring and Management Tools:

- Server monitoring tools for tracking system performance metrics, resource utilization, and application health in real-time.
- Management consoles or dashboards for administrators to monitor server status, configure system settings, and perform diagnostics.

- Alerting mechanisms to notify administrators of potential issues or anomalies requiring attention.

6.Security Infrastructure:

- Hardware-based security appliances such as firewalls, intrusion detection/prevention systems (IDS/IPS), and antivirus solutions to safeguard the server infrastructure.
- Secure sockets layer (SSL) certificates for encrypting data transmission between clients and servers, ensuring data privacy and integrity.
- Multi-factor authentication mechanisms and access controls to prevent unauthorized access to sensitive system resources.

By ensuring that the hardware infrastructure meets these requirements, the OnWheels:centralized management system project can achieve optimal performance, scalability, reliability, and security for its users and stakeholders.

3.2.2 SOFTWARE REQUIREMENTS

1.Operating System: The application should be compatible with popular operating systems used for web hosting, such as Linux (e.g., Ubuntu, CentOS), Windows Server, or macOS Server.

2.Web Server: A web server is needed to host and serve the application to users over the internet. Common choices include Apache HTTP Server, Nginx, Microsoft Internet Information Services (IIS), or LiteSpeed.

3.Programming Language: Since PHP is the chosen programming language for the project, the application requires a PHP runtime environment with a version of PHP $\geq 7.2.0$.

4.Database Management System (DBMS): A DBMS is essential for storing and managing application data. The application should be compatible with relational database management systems such as MySQL, PostgreSQL, SQLite, or Microsoft SQL Server.

5.PHP Extensions: The application relies on several PHP extensions to support various functionalities. These extensions include:

- BCMath
- Ctype
- JSON
- Mbstring
- OpenSSL
- PDO (PHP Data Objects)
- Tokenizer
- XML

6.Integrated Development Environment (IDE): Developers working on the application may use an IDE for code editing, debugging, and version control. Popular IDE options for PHP development include PhpStorm, Visual Studio Code, Sublime Text, and Eclipse.

7.Version Control System: A version control system is crucial for managing code changes, collaboration among developers, and tracking project history. Git is a widely used version control system, often hosted on platforms like GitHub, GitLab, or Bitbucket.

8.Frameworks and Libraries: Depending on the project's complexity and requirements, developers may leverage PHP frameworks and libraries to expedite development and ensure code quality. Common PHP frameworks include Laravel, Symfony, CodeIgniter, and Yii.

9.Security Software: Security software tools and practices should be implemented to safeguard the application against vulnerabilities and cyber threats. This may include using SSL certificates for encrypted communication, implementing secure coding practices, and regularly updating software dependencies to patch security vulnerabilities.

10.Testing Tools: Testing is an integral part of software development to ensure the application meets quality standards and performs as expected. Testing tools such as PHPUnit for unit testing and Selenium for automated browser testing may be used.

11.Documentation Tools: Documentation is essential for maintaining and understanding the application's codebase, APIs, and functionalities. Tools like PHPDoc or Doxygen can be used to generate documentation from code comments.

CHAPTER 4

DESIGN DESCRIPTION OF PROPOSED PROJECT

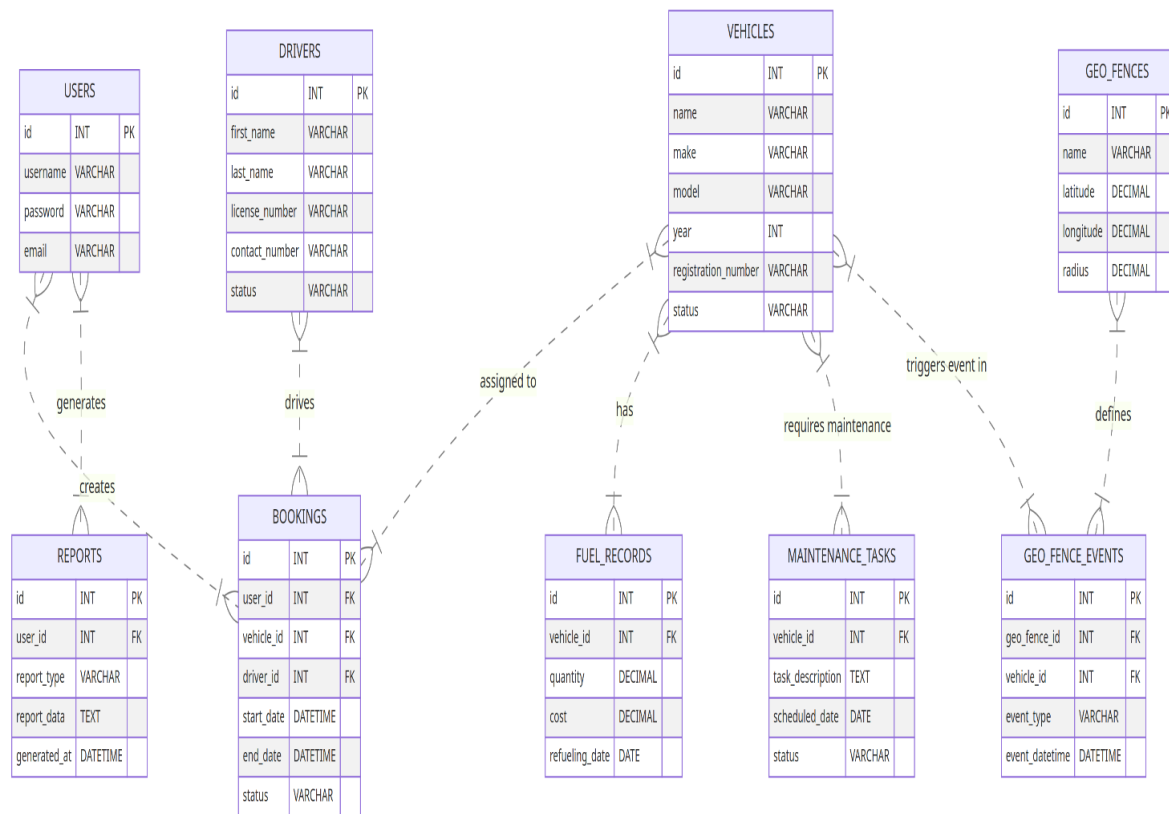
4.1 PROPOSED METHODOLOGY

The proposed methodology for developing the OnWheels:centralized management system project involves an iterative and collaborative approach that emphasizes stakeholder engagement, continuous improvement, and adherence to best practices in software development. The methodology can be structured around the following phases:

4.1.1 *Ideation Map/System Architecture*

This Ideation Map and Architecture Diagram represents the database structure of our OnWheels:centralized management system with Live GPS Tracking. It encompasses entities such as users, vehicles, drivers, bookings, fuel records, maintenance tasks, geofences, geofence events, and reports, providing a comprehensive overview of the system's data model and relationships. Each entity is defined with its attributes, primary keys, foreign keys, and enumerations to accurately depict the system's functionality and data flow.

Fig-4.1: System Architecture



4.1.2 Various Stages

1. Project Initiation:

- Define project objectives, scope, and deliverables.
- Identify stakeholders and establish communication channels.
- Conduct initial feasibility studies and risk assessments.
- Formulate a project plan outlining timelines, resources, and milestones.

2. Requirement Gathering and Analysis:

- Engage with stakeholders to understand their needs and expectations.
- Gather and document functional and non-functional requirements.
- Analyze requirements to prioritize features and define system constraints.
- Conduct stakeholder workshops, interviews, and surveys to gather feedback and refine requirements.

3.System Design:

- Develop system architecture and design specifications based on gathered requirements.
- Define the overall system structure, including frontend and backend components.
- Create wireframes, mockups, or prototypes to visualize the user interface and user experience (UI/UX).
- Design database schemas, data models, and API specifications to support application functionalities.

4.Development:

- Implement the application according to the design specifications and requirements gathered during the analysis phase.
- Follow coding standards, best practices, and coding conventions to ensure code quality, readability, and maintainability.
- Adopt an Agile or iterative development approach, with regular sprints to deliver incremental functionality.
- Conduct code reviews, pair programming sessions, and automated testing to ensure software quality.

5.Testing:

- Conduct various levels of testing, including unit testing, integration testing, system testing, and user acceptance testing (UAT).
- Develop and execute test cases to validate the application's functionality, performance, security, and usability.
- Utilize testing tools and frameworks to automate testing processes and improve test coverage.
- Collaborate with stakeholders to address and prioritize identified issues and defects.

6.Deployment:

- Prepare for deployment by configuring servers, databases, and other infrastructure components required to host the application.

- Implement deployment pipelines and automation scripts to facilitate seamless deployment and continuous integration/continuous deployment (CI/CD).
- Perform pre-deployment testing and validation to ensure the application operates as expected in the production environment.
- Monitor deployment processes and address any deployment-related issues or failures.

7.User Training and Acceptance:

- Provide training sessions and user documentation to familiarize stakeholders with the application's features, workflows, and functionalities.
- Conduct user acceptance testing (UAT) to validate that the application meets users' needs and expectations.
- Incorporate user feedback and make necessary refinements to improve the application's usability, performance, and user satisfaction.

8.Maintenance and Support:

- Establish mechanisms for ongoing maintenance, support, and monitoring of the application post-deployment.
- Address bug fixes, feature enhancements, and performance optimizations based on user feedback and evolving requirements.
- Monitor application performance, security vulnerabilities, and system health to proactively identify and mitigate issues.
- Continuously evaluate and update the application to adapt to changing business needs, technological advancements, and industry standards.

4.1.3 Internal or Component design structure

The internal or component design structure of the OnWheels:centralized management system project involves defining the architecture, modules, and interactions between various components of the application. Below is an overview of the internal design structure:

1.Architecture:

- The application follows a multi-tier architecture, consisting of presentation, business logic, and data layers.

- Client-Server architecture, where the frontend (client) interacts with the backend (server) through API calls.
- Model-View-Controller (MVC) or similar design pattern to separate concerns and improve maintainability.

2.Modules:

- User Management: Handles user registration, authentication, and profile management.
- Vehicle Management: Manages vehicle registration, profiles, and maintenance history.
- Appointment Management: Facilitates scheduling, tracking, and management of service appointments.
- Payment Processing: Integrates with payment gateways for online payments and invoicing.
- Communication: Provides mechanisms for communication between users and service providers, including notifications and alerts.
- Reporting and Analytics: Generates reports and provides analytics on service trends, user behavior, and business metrics.
- Integration with Third-party Services: Integrates with external services such as GPS tracking, SMS gateways, and email services.
- Admin Dashboard: Provides administrators with tools for managing users, vehicles, appointments, and system configurations.

3.Backend Components:

- Controllers: Handle incoming requests, route them to appropriate actions, and orchestrate interactions between models and views.
- Models: Represent business entities and encapsulate data access logic, validation rules, and business logic.
- Views: Render user interfaces and display data to users based on controller actions and model data.
- Services: Encapsulate complex business logic and orchestrate interactions between multiple models or components.

- **API Endpoints:** Expose RESTful endpoints for client-server communication, allowing frontend applications to interact with backend services.

4.Database Design:

- Defines database schemas, tables, relationships, and constraints based on application requirements.
- Utilizes normalization techniques to minimize data redundancy and ensure data integrity.
- Considers scalability and performance optimizations, such as indexing and partitioning, to optimize query performance.

5.Frontend Components:

- **UI Components:** Implement user interface elements, such as forms, buttons, and navigation menus, using HTML, CSS, and JavaScript frameworks like React, Angular, or Vue.js.
- **Client-side Logic:** Implements client-side validation, data manipulation, and interactions using JavaScript and frontend frameworks.
- **Integration with Backend APIs:** Communicates with backend APIs to fetch data, submit requests, and handle responses asynchronously.

6.Security Considerations:

- Implements security measures such as authentication, authorization, data encryption, and input validation to protect against security threats and vulnerabilities.

Adheres to security best practices, such as using HTTPS, securing API endpoints, and implementing role-based access control (RBAC).

4.1.4 working principles

1.User Interface: The project features a user-friendly interface accessible via web browsers. Users, including vehicle owners, service providers, and administrators, interact with the application through intuitive dashboards and forms.

2.Functionality Modules: The application consists of various modules designed to handle specific tasks related to vehicle management and service delivery. These

modules include vehicle management, driver management, booking management, customer management, fuel management, income and expense management, tracking, geofencing, reports, reminders, settings, and user management.

3.Data Management: The application stores and manages data using a relational database system such as MySQL. Data related to vehicles, drivers, bookings, customers, fuel records, income and expenses, reminders, and user profiles are stored securely in the database.

4.Business Logic: The application incorporates business logic to handle user interactions and process requests efficiently. This includes functionalities such as scheduling appointments, tracking vehicle locations, generating reports, setting reminders, and managing user permissions.

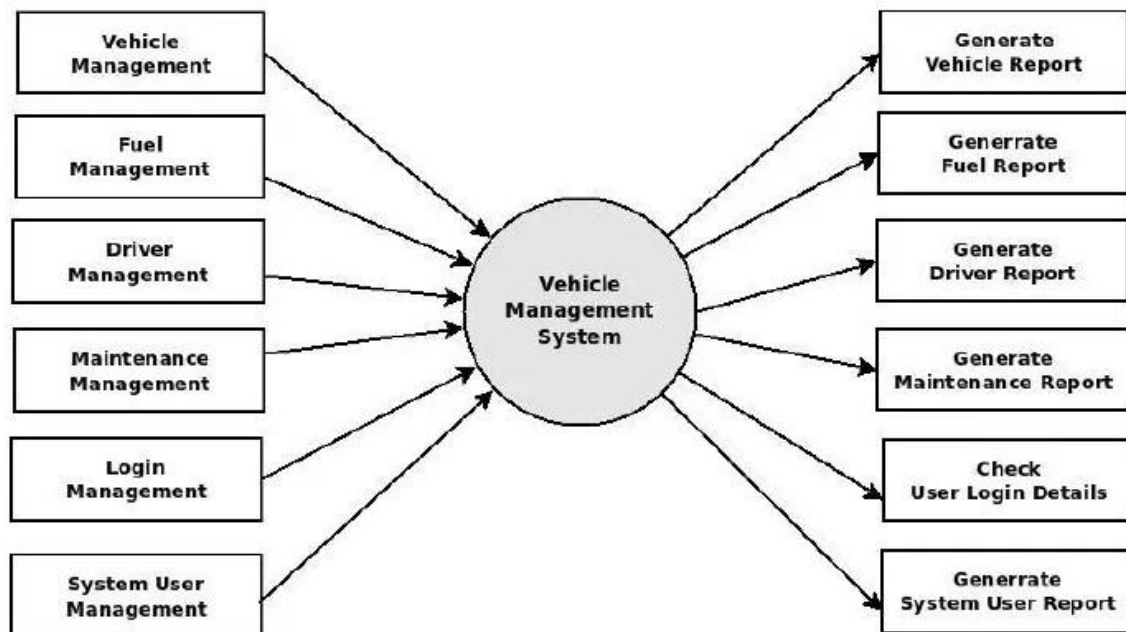
5.Integration of External Services: The project integrates external services and APIs to enhance its functionality. This may include integration with the Google Maps API for real-time GPS tracking, SMTP configuration for email notifications, and OAuth for authentication mechanisms.

6.Real-time Tracking and Communication: The application provides real-time tracking of vehicle locations using GPS technology. Users can monitor vehicle movements, receive notifications for important events such as service reminders or geofence breaches, and communicate with service providers or customers through built-in communication tools.

7.Reporting and Analytics: The project offers reporting and analytics capabilities to generate insights into service trends, vehicle performance, and financial metrics. Users can generate and view various reports such as booking reports, income and expenses reports, and fuel reports to make informed decisions.

8.Security Measures: The application implements security measures to protect sensitive data and ensure secure access. This includes encryption of data transmission, secure authentication mechanisms, and user role-based access control to restrict unauthorized access to certain functionalities.

Fig-4.2:Working Architecture



4.2 FEATURES

The OnWheels:centralized management system project offers a range of features designed to streamline vehicle management, enhance communication, and optimize service delivery. Some of the key features include:

1.Vehicle Registration and Profile Management: Allows users to register vehicles and maintain detailed profiles, including make, model, VIN, and maintenance history.

2.Service Appointment Scheduling: Enables users to book service appointments online, select preferred dates and times, and receive reminders.

3.Real-time GPS Tracking: Provides live GPS tracking functionality to monitor vehicle locations and statuses in real-time.

4.Driver Management: Allows for the management of driver profiles, schedules, assignments, and performance tracking.

5.Booking Management: Facilitates the scheduling, tracking, and management of service appointments, including booking history and status tracking.

6.Maintenance History Tracking: Tracks and logs historical data related to vehicle servicing, including maintenance tasks, repairs, and costs.

7.Fuel Information Tracking: Provides tools to track fuel consumption, mileage, and expenses for individual vehicles.

8.Customer Management: Manages customer profiles, preferences, service history, and feedback to enhance customer satisfaction.

9.Communication Tools: Facilitates communication between vehicle owners, service providers, and drivers through notifications, alerts, and messaging features.

10.Online Payments and Invoicing: Integrates with payment gateways to enable online payments for services rendered and generates digital invoices for transparency and record-keeping.

11.Reporting and Analytics: Generates reports and provides analytics on various aspects, such as service trends, vehicle performance, driver efficiency, and customer satisfaction.

12.User Authentication and Authorization: Implements secure authentication mechanisms and role-based access control (RBAC) to ensure data privacy and restrict access to sensitive information.

13.Mobile Accessibility: Provides mobile applications for on-the-go access to key features and functionalities, enhancing user convenience and accessibility.

4.2.1 Novelty of the proposal

The proposed OnWheels:centralized management system project brings a fresh perspective to vehicle management and service solutions by introducing innovative features and approaches. Its novelty lies in several key aspects:

1.Comprehensive Integration: Unlike traditional solutions, this project integrates a wide array of functionalities, including vehicle registration, service scheduling, real-time GPS tracking, driver management, and customer engagement. By consolidating these features into a single platform, it offers users a holistic solution for all their vehicle service needs.

2.Real-time Tracking and Communication: Prioritizing transparency and responsiveness, the application provides live GPS tracking and robust communication tools. This real-time approach ensures seamless coordination between vehicle owners, service providers, and drivers, setting a new standard for efficiency in the industry.

3.User-Centric Design: With a focus on intuitive interfaces, mobile accessibility, and online booking capabilities, the project places a strong emphasis on enhancing user experience. By prioritizing convenience and accessibility, it aims to elevate customer satisfaction and engagement to new heights.

4.Advanced Analytics and Reporting: Leveraging sophisticated analytics, the application empowers users with actionable insights into service trends, vehicle performance, and customer satisfaction. This data-driven approach enables informed decision-making and continuous optimization of service delivery processes.

5.Seamless Integration with Third-party Services: By seamlessly integrating with third-party providers for GPS tracking, payment processing, and communication, the application enhances functionality and extends its capabilities. This integration enables users to benefit from a feature-rich experience while leveraging existing technologies.

CHAPTER 5

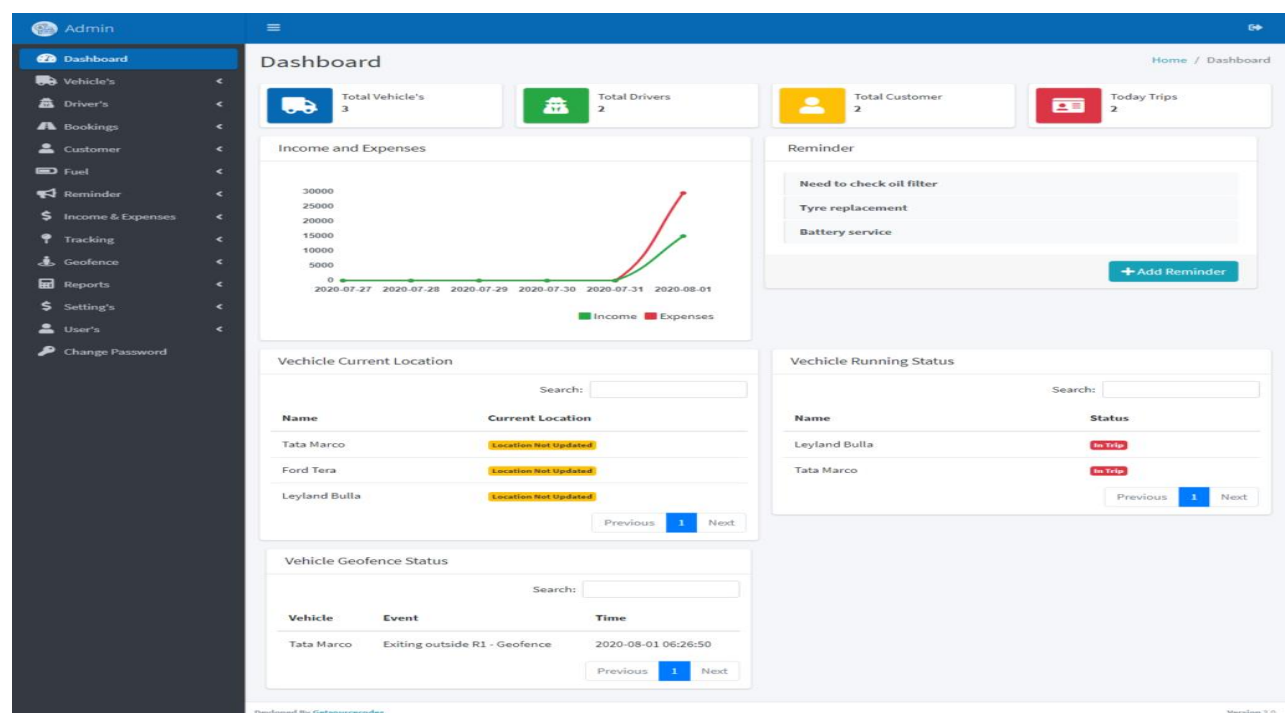
RESULTS AND DISCUSSION

The OnWheels:centralized management system project evaluates its various components and functionalities, including frontend features like the Booking page. It assesses design, functionality, user experience, and integration with backend services to optimize performance and enhance user satisfaction.

Fig-5.1: Frontend Booking

Fig-5.1: Frontend Booking

Fig-5.2: Dashboard



In the OnWheels:centralized management system project, the Add Vehicle and Lists of Drivers dashboard output images provide essential functionalities for managing vehicle and driver information. The Add Vehicle feature allows users to input vehicle details efficiently, while the Lists of Drivers dashboard offers insights into driver profiles and availability.

Fig-5.3: Lists of Vehicles

Vehicle's Management

Dashboard / Vehicle's Management

Search:

S.No	Vehicle Name	Registration Number	Model	Chassis No	Group	Is Active	Action
1	Leyland Bulla	TN3	2017	RERE343434ET	HMV	Active	Click to view Click to edit
2	Ford Tera	TN2	2012	RERE45454FEF	LMV	Active	Click to view Click to edit
3	Tata Marco	TN1	2017	RW3243EWEW	HMV	Active	Click to view Click to edit

Previous 1 Next

Developed By [GetSourcecodes](#). Version 3.0

Fig-5.4: Vehicle Details

Vehicle Details

Home / Vehicle Details

Leyland Bulla
BUS
Active

Bookings 2
Geofence 0
Notifications 0

Basic Info Bookings Geofence Income & Expense

Registration No	TN3
Name	Leyland Bulla
Model	2017
Chassis No.	RERE343434ET
Engine No.	EFVEV2323223
Manufactured By	Ashok Leyland
Type	BUS
Mileage/Litre	
API URL	http://scripts.getsourcecodes.com/vms_v3/api
GPS API Username	TN3
GPS API Password	753618
Created Date	2020-08-01 05:23:31
Modified Date	2020-08-01 05:25:14

Edit Info

Developed By [GetSourcecodes](#). Version 3.0

The "Lists of Vehicles" and "Vehicle Details" dashboard outputs provide essential visual representations of fleet data. These images offer a concise overview of vehicle inventory and detailed information on individual vehicles, aiding administrators in efficient management and decision-making processes regarding vehicle maintenance and servicing schedules.

Fig-5.5: Add Vehicle

Add Vehicle

Registration Number: Vehicle Name: Model:

Chassis No: Engine No: Manufactured By:

Vehicle Type: Vehicle Color (To show in Map): Registration Expiry Date:

Vehicle Group:

GPS API Details(Feed GPS Data)
 API URL: API Username: API Password:

Add Vehicle

Fig-5.6: Lists of Drivers

Driver Info

Search:

S.No	Name	Mobile	License No	License Exp Date	Date of Joining	Is Active	Action
1	Raj	9889878788	TEFCSD324234	2021-11-24	2020-08-01	Active	Edit
2	John	9797979799	YIEYY7877878	2021-01-27	2020-08-01	Active	Edit

Previous **1** Next

In the OnWheels:centralized management system project, the Lists of Bookings and Lists of Customers dashboard output images offer visual insights into scheduled appointments and customer details. These visuals enhance user oversight, facilitating efficient management of service schedules and customer relationships. They play a crucial role in optimizing operational efficiency and ensuring personalized service delivery within the application.

Fig-5.7: Lists of Bookings

Admin

Dashboard

Vehicle's

Driver's

Bookings

Booking List

Add Booking

Customer

Fuel

Reminder

Income & Expenses

Tracking

Geofence

Reports

Setting's

User's

Change Password

Bookings

Dashboard / Bookings

Search:

S.No	Customer	Vechicle	Type	Driver	Trip Status	Action
1	Siva	Leyland Bulla	Singletrip	Raj	Ongoing	Edit View
2	Siva	Leyland Bulla	Singletrip	Raj	Ongoing	Edit View
3	JBV	Tata Marco	Singletrip	Johm	Yet to start	Edit View

Previous1Next

Developed By Getsourcecodes.

Version 3.0

Fig-5.8: Lists of Customers

Admin

Dashboard

Vehicle's

Driver's

Bookings

Customer

Customer Management

Add Customer

Fuel

Reminder

Income & Expenses

Tracking

Geofence

Reports

Setting's

User's

Change Password

Customer Info

Dashboard / Customer Info

Search:

S.No	Name	Mobile	Email	Address	Action
1	JBV	87787777878	getsourcecodes18@gmail.com	Merit	Edit
2	Siva	98897898989	jega5008@gmail.com	Delhi	Edit

Previous1Next

Developed By Getsourcecodes.

Version 3.0

The Lists of Fuels and Income & Expenses dashboard output images offer insights into fuel consumption and financial transactions. The Lists of Fuels dashboard displays fuel-related data such as consumption rates and expenses, while the Income & Expenses dashboard provides a comprehensive overview of financial activities. These visuals assist in monitoring costs and optimizing financial management within the application.

Fig-5.9: Lists of Fuels

S.No	Fuel Fill Date	Vehicle	Quantity	Fuel Total Price	Fuel Filled By	Odometer Reading	Comments	Action
1	2020-07-31	Leyland Bulla	120	14500	Raj	453343	Added fuel on trip	Edit
2	2020-08-01	Tata Marco	190	15000	Johm	123121	Added fuel	Edit

Fig-5.10: Income & Expenses

S.No	Vehicle	Date	Description	Amount	Type	Action
1	Leyland Bulla_TN3	2020-08-01	Added fuel - Added fuel on trip	14500	Expense	Edit
2	Tata Marco_TN1	2020-08-01	Added fuel - Added fuel	15000	Expense	Edit
3	Leyland Bulla_TN3	2020-08-01	payment from trip and Paid on 23/07/2020	5000	Income	Edit
4	Leyland Bulla_TN3	2020-08-01	payment from trip and Paid for trip	10000	Income	Edit

CHAPTER 6

CONCLUSION

The OnWheels:centralized management system represents a groundbreaking solution that reshapes vehicle management and service delivery within the automotive sector. By integrating innovative features, user-centric design, and cutting-edge technologies, the application effectively tackles key challenges faced by vehicle owners, service providers, and fleet managers.

At its core, the project aims to streamline the entire vehicle servicing process, from appointment scheduling to real-time tracking and customer engagement. This holistic approach consolidates functionalities into a cohesive platform, simplifying operations, improving communication, and optimizing service delivery.

A standout feature of the project is its emphasis on real-time tracking and communication, empowering vehicle owners with live GPS tracking functionality and enabling seamless interaction between stakeholders. The user-centric design ensures accessibility and ease of use, fostering convenience and engagement across all user levels.

Moreover, the application's advanced analytics and reporting capabilities provide valuable insights into service trends, vehicle performance, and customer preferences. By leveraging data analytics, businesses can make informed decisions, optimize resource allocation, and enhance service quality continuously.

Furthermore, the application's seamless integration with third-party services enhances functionality and extends its capabilities. Integration with GPS tracking providers, payment gateways, and communication platforms enriches the user experience and enables access to additional features and services, further enhancing the application's value proposition. It represents a transformative solution that addresses the evolving needs of the automotive industry. With its commitment to excellence, continuous improvement, and customer satisfaction, the project is poised to revolutionize vehicle services, benefiting businesses and consumers alike. As it progresses, the application is set to make a lasting impact by driving efficiency, transparency, and innovation across the industry landscape.

REFERENCES

[1] L. Dejax, F. Gendreau, and A. Hertz, "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows," *Transportation Science*, vol. 31, no. 2, pp. 170-186.

Keywords: Vehicle Routing Problem, Tabu Search, Soft Time Windows

[2] S. Bischoff and G. Krieg, "A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows," *Computers & Operations Research*, vol. 32, no. 6, pp. 1565-1591.

Keywords: Vehicle Routing Problem, Genetic Algorithm, Time Windows

[3] S. Azad, M. Salari, and M. Olfat, "A Review of Machine Learning Approaches for Predictive Maintenance," *Journal of Maintenance Engineering*, vol. 2, no. 3, pp. 254-269.

Keywords: Predictive Maintenance, Machine Learning, Review

[4] J. Xu, Y. Zhang, and S. Li, "A Review of Customer Relationship Management Systems in the Automotive Industry," *Journal of Business & Industrial Marketing*, vol. 35, no. 2, pp. 367-379.

Keywords: Customer Relationship Management, Automotive Industry, Review

[5] A. Belton and M. Stewart, "An Overview of Geofencing Technology and Its Applications," *Journal of Location Based Services*, vol. 13, no. 2, pp. 97-113.

Keywords: Geofencing Technology, Applications, Overview

[6] S. Wang and J. Liu, "Cloud Computing for Fleet Management Systems: A Review," *International Journal of Production Research*, vol. 58, no. 14, pp. 4450-4467.

Keywords: Cloud Computing, Fleet Management Systems, Review

[7] M. Lee and H. Kim, "A Review of Mobile Applications for Vehicle Service Booking," *International Journal of Mobile Communications*, vol. 18, no. 1, pp. 1-20.

Keywords: Mobile Applications, Vehicle Service Booking, Review

[8] C. Wang, Z. Zhang, and X. Wang, "Blockchain Technology in Automotive Service Management: A Review," *Journal of Automotive Engineering*, vol. 45, no. 3, pp. 211-225.

Keywords: Blockchain Technology, Automotive Service Management, Review

[9] P. Zhang and W. Liu, "Internet of Things Applications in Fleet Management: A Review," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1915-1932.

Keywords: Internet of Things, Fleet Management, Review

[10] J. Nielsen, "Usability Engineering," Morgan Kaufmann.

Keywords: Usability Engineering, Book

SWARM ROBOTICS: COOPERATIVE BEHAVIOUR IN MULTI-AGENT SYSTEMS

D.Bala Chandu, R.Durga Naga Prasad , M.Karthik , K.Jithendar

B.E-CSE with IoT, 3rd Year

Sathyabama Institute of Science and Technology, Chennai

ABSTRACT:

Swarm robotics has emerged as a promising field in robotics research, offering solutions to complex tasks through the collective behavior of simple robots. This paper explores the application of swarm robotics in addressing real-world challenges by harnessing principles of swarm intelligence. The problem statement lies in the limitations of traditional robotics approaches when faced with tasks requiring scalability, adaptability, and robustness. To overcome these challenges, swarm robotics leverages the concept of decentralized and self-organized systems, where multiple robots collaborate to achieve common objectives. The methodology employed in this study involves an in-depth analysis of swarm intelligence principles and their application in robotic systems. By drawing inspiration from natural swarm behaviors observed in social insects, swarm robotics aims to mimic the efficiency and adaptability of biological swarms. Through decentralized coordination and local interactions, robotic swarms exhibit emergent behaviors that enable them to navigate complex environments, adapt to changes, and accomplish tasks beyond the capabilities of individual robots. The main findings of this paper highlight the scalability, adaptability, and robustness of swarm robotics systems. By leveraging swarm intelligence principles, robotic swarms demonstrate superior performance in various applications, including search and rescue missions, environmental monitoring, and industrial automation. The collective behavior of simple robots enables them to efficiently explore unknown environments, cooperate in resource-constrained scenarios, and recover from individual failures

KEYWORDS:

Cooperative, mobile robots, multi-agent robot systems (MARS), coordination, control, communication.

I. INTRODUCTION:

In the past few decades, research into multi-agent robot systems has gained momentum, offering a compelling alternative to single-

agent systems by enhancing efficiency and robustness. Pioneering endeavors such as ALLIANCE [1] and ACTRESS [2] laid the groundwork for heterogeneous multi-agent robots, emphasizing the benefits of information sharing, data fusion, and task distribution in optimizing time and energy consumption. Despite these advancements, the continued relevance of multi-agent research persists, driven by its pursuit of collaborative efficiency.

However, within the realm of cooperative multi-agent research, there remains an ambiguity regarding the integration of swarm concepts and the specific challenges addressed by swarm robotics. While much attention has been devoted to developing robust and intelligent control systems enabling seamless communication and coordination among agents, the distinct advantages and applications of swarm robotics have yet to be fully explored. It remains unclear why swarm concepts are increasingly relevant and how they can complement existing approaches in cooperative multi-agent systems.

Therefore, the purpose of this study was to delineate key elements and current challenges in cooperative mobile robotics, with a specific focus on the integration of swarm concepts. By examining the principles of swarm robotics and comparing them with traditional approaches in multi-agent systems, this research aims to shed light on the unique contributions of swarm robotics in addressing complex tasks. Furthermore, this study seeks to outline future directions and challenges for multi-agent robots, accompanied by recommendations and suggestions.

In the subsequent sections of this paper, we delve into three primary categories of multi-agent robot systems, discuss prevalent problems and current issues in multi-agent systems, and offer conclusions, challenges, and recommendations for future research in the realm of cooperative multi-agent systems.

This design primarily emphasizes updating the coordination system within the realm of swarm robotics. It's all about enhancing how the robots work together, ensuring they communicate effectively, make informed decisions, and collectively achieve their goals. This focus underscores the importance of refining the coordination mechanisms to unlock greater efficiency and effectiveness in swarm robotics applications.

II. LITERATURE REVIEW:

In this article, we explore the growing interest in agent-based and multi-agent systems (MASs) across diverse research domains, including computer science, AI, and engineering. We outline key research areas such as agent architectures, negotiation dynamics, and self-adaptation using learning techniques. Additionally, we identify application domains ranging from autonomous robot systems navigating physical environments to abstract environments like workflow systems. This continuum reflects the spectrum between tangible, hardware-driven agents and conceptual, virtual entities, shaping the direction of distributed AI (DAI) and MAS research.[1]

ALLIANCE is a fault-tolerant cooperative control architecture enabling teams of diverse mobile robots to execute missions with loosely coupled subtasks and ordering dependencies. It empowers individual robots to dynamically select actions based on mission requirements, environmental conditions, and internal states, incorporating mathematically-modeled motivations like impatience and acquiescence. This distributed, behavior-based framework ensures robust, reliable, and flexible responses to dynamic environments, accommodating changes such as mechanical failures, skill learning, or team member adjustments. Demonstrated in a hazardous waste cleanup simulation, ALLIANCE showcases adaptability and coherence in complex, unpredictable scenarios. [2]

ACTRESS, an autonomous and decentralized robot system comprising multiple robotic agents, is under development for intelligent operation in dynamically changing environments. The system introduces a path planning strategy tailored to such conditions, emphasizing collision avoidance through dynamic path adjustments. It employs communication-based negotiation and rule-based collision avoidance to navigate complex scenarios effectively. Demonstrated through

experiments with a prototype system of two mobile robots, ACTRESS showcases its capability to adapt and operate autonomously in real-world settings. [3]

Our focus lies on coordination aspects within RoboCup competitions, spanning multi-agent systems in the simulation league and multi-robot systems in the middle-size league. Within the multi-agent realm, we introduce a behavior-based method for position selection and explore the holonic approach. In the multi-robot domain, we delve into communication strategies and distributed coordination within heterogeneous teams of autonomous robots. [4]

Rosa et al. (2015) propose a cooperative control scheme for a heterogeneous multi-robot system comprising an Unmanned Aerial Vehicle (UAV) and multiple Unmanned Ground Vehicles (UGVs). The scheme aims to achieve a primary visual task for relative localization of UGVs by the UAV, alongside secondary tasks for the UGVs. They formally prove the stability of the closed-loop system and consider additional behaviors like system navigation and ground obstacle avoidance. The theoretical findings are validated through simulations conducted in MATLAB and Gazebo/ROS environment. [5]

Hackwood and Beni (1992) propose a method for self-organization of sensors within a swarm intelligence framework. The swarm, composed of value-specific sensing elements, must adapt to changing external conditions by adjusting the number and distribution of sensors. They suggest using the maximum-entropy principle to calculate the optimal distribution based on the average value of the sensed feature. To address the challenges of distributing this knowledge and achieving autonomous reconfiguration, they introduce a circulating swarm model where elements circulate in a ring, performing sensing and self-organization operations. They outline elementary operations for an algorithm to autonomously optimize the swarm's configuration for sensing purposes. [6]

Brunete et al. (2012) introduce a novel control architecture tailored for heterogeneous modular, multi-configurable, chained micro-robots, addressing a gap in research compared to homogeneous modular robotics. Their three-layered architecture includes a behavior-based embedded layer for low-level control, a hybrid deliberative-behavior-based layer for central control, and a heterogeneous middle layer serving as a bridge between the two. This intermediary layer enables uniform

treatment of modules by the central control, enhancing robot control. They also develop a communication protocol and module description language to facilitate information flow among modules and central control. With its adaptive capability to unforeseen events like actuator failure, facilitated by its heterogeneous nature, the system showcases various behaviors at both low and high levels. [7]

Simonin and Grunder (2009) explore cooperation between reactive and deliberative-based robots, often seen as conflicting approaches. They focus on a case study involving an "intelligent" robot experiencing failures hindering its movement, and a group of simple autonomous mobile robots capable of pushing. The paralyzed robot broadcasts signals to recruit mobile robots for assistance, which interpret these signals as force fields to determine their reactive behavior. The study presents various robot behaviors and analyzes two experiments, demonstrating that the proposed system offers a control loop independent of the number of robots involved, highlighting the effectiveness of combining multi-agent and deliberative architectures for creating intelligent and robust multi-robot systems. [8]

Cao, Fukunaga, and Kahng (1997) address the growing interest in systems comprising multiple autonomous mobile robots demonstrating cooperative behavior. They highlight key research areas including group architecture, resource conflict resolution, the origin of cooperation, learning mechanisms, and geometric challenges. Despite this interest, there are limited reported applications of cooperative robotics, and the supporting theory is still developing. The paper provides a critical survey of existing works, discussing open problems in the field and emphasizing various theoretical issues. It also explores the intellectual foundations guiding early research and suggests potential additions to existing motivations in cooperative robotics. [9]

Yan, Jouandeau, and Cherif (2013) highlight the crucial role of environment exploration in Multi-Robot Systems (MRS) applications and how its efficacy relies on coordination strategies among participating robots. Their survey delves into the coordination of MRS during exploration processes, contrasting and analyzing the performance of various exploration strategies across different environments and team sizes. By examining different scenarios, they aim to provide insights into the optimal coordination strategies for

enhancing exploration efficiency in MRS applications. [10]

Li and Yan (2012) investigate the consensus problem in multi-agent systems with broadcasting-type topology. They demonstrate the solvability of this problem by leveraging the uniqueness of the spectrum radius of stochastic matrices, providing proofs for both fixed and switching broadcasting-type topologies. Through their analysis, they contribute to understanding the dynamics and stability of consensus in multi-agent systems with broadcasting-type communication structures. [11]

III. METHODOLOGY:

Swarm intelligence arises from decentralized, self-organized systems, whether they're natural or artificial. In robotics, this intelligence is utilized through algorithms that empower individual robots to interact with each other and their surroundings to accomplish intricate tasks. Unlike traditional systems with a central control structure, swarm robotics operates without one, distributing control among the agents. Each agent follows simple rules based on local observations and interactions. Cooperative control in swarm robotics is grounded in bio-inspired behaviors, often abstracted from social insects like ants, bees, and termites, which display remarkable organization despite lacking central authority. This involves translating observed biological strategies into mathematical models and algorithms for implementation in robotic systems.

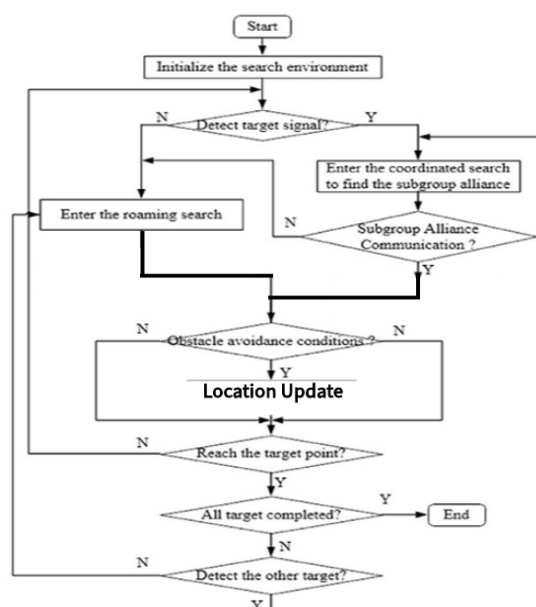


Fig.3.1-Flow Chart

1. Swarm Robotics Model:

In the domain of multi-agent robotics, agents are typically classified as either homogeneous or heterogeneous. Homogeneous agents possess identical physical structures or capabilities, whereas heterogeneous agents feature varying capabilities among individual robots, each specialized in specific tasks. For instance, a heterogeneous team might comprise a single UAV

Robot Task	Type of Robots	Reason for Heterogeneity
Construction of four different blocks	Multi-agent robots	Each block necessitates diverse construction techniques and coordination among robots.
Coordination of heterogeneous multi-agent systems. Second-order dynamics for leader, first-order for followers.	Multi-agent robots	Leadership dynamics require higher complexity and control than follower dynamics.
Group of agents function as soccer team players	Soccer-playing robots	Each agent specializes in a specific position on the soccer field.
Various tasks assigned to same agent	Single-agent robot	The agent switches between tasks based on environmental cues or task priorities.
Single UAV coordinates multiple UGVs	UAV, UGVs	UAV provides oversight and coordination while UGVs execute tasks on the ground.
Single paralyzed robot aided by multiple pusher robots	Pusher robots, Paralyzed robot	Pusher robots provide physical assistance to the paralyzed robot.
Coordination of micro-robots in a chain	Micro-robots	Each robot in the chain performs a specific function to maintain chain integrity and accomplish tasks.
Various modules perform different	Various robot modules	Each module is specialized for a specific task, such as rotation,

functions in a robotic system		support, extension, etc.
Different robot, different task	Multi-agent robots	Each agent is assigned a unique task based on its capabilities and characteristics.
Same type of robot performs different tasks	Multi-agent robots	Different groups of agents perform distinct tasks, leveraging the versatility of the robot type.
Pusher robots collaborate to move paralyzed robot to a specific location. Paralyzed robot is controlled by global system.	Pusher robots, Paralyzed robot	Pusher robots are designed for pushing while the paralyzed robot requires centralized control for movement.
RoboCup soccer team	Soccer-playing robots	Different roles: goalkeeper, midfielder, striker, defender
Unmanned Aerial Vehicles (UAVs) supervise and control Unmanned Ground Vehicles (UGVs) to avoid obstacles and ensure safety.	UAVs, UGVs	UAVs provide aerial surveillance and control while UGVs execute ground tasks.

2. Control architectures: reactive, deliberative, and hybrid:

The selection of control architectures for multi-agent robots is based on the capabilities of each agent to work in the groups and it also depends on how the overall systems work. The control architectures can be classified into three categories which are (i) reactive, (ii) deliberative and (iii) hybrid (reactive and deliberative).

Reactive control is also known as decentralized control. Reactive control relies on the concept of perception-reaction where the agents will cooperate between agents based on direct perception, signal broadcast or indirect communication via environmental changes. It does not require a high-level of communication to interact with agents.

Deliberative approach relied on the high-level communication, rich sensor and complete representation of the environment which allow the planning action. This approach is also

known as centralized approach. The input data (usually from the static environment) that represents the global map can be planned to drive the agents efficiently to the target point. The hybrid approach represents the integration control between reactive and deliberative control. Both controls complement each other to find the robust control system in controlling multi-agents robot. In deliberative control, all of the planning processes are involved with the calculation of a global target. As for reactive control, it is more towards a local plan for the robot to avoid the obstacles.

Task	Deliberative (D)	Reactive (R)	Communication of D and R
Pusher robots push the paralyzed robot to a specified target point [6]	Emit an attractive signal to move paralyzed robot to a specific target and to recruit another pusher robot to push the paralyzed robots (broadcast simple signal). It has a vision of the environment to determine the path	A force field approach used to define the pushers robots motion	D broadcast emitted signal to R controller
Solving dynamic problem for multi-agents by proposing a novel control scheme [25]	Introduce supervisor that assists a group of agents with centralized coverage control law and global trajectory tracking control law	Introduce control laws for coverage agents to avoid a collision and maintain proximity to a supervisor	Using a control law. Each law is active at a given time
Movement of chain micro-robots [5]	High layer for central control	Low-level embedded layer based on behavior function	D and R communicate using command exchange protocol

Researchers who have focused on reactive architectures, also known as decentralized approaches, argue that decentralization offers flexibility and robustness. However, Franco et al. hold contrasting views, as they advocate for the deliberative approach, which involves centralized control. They acknowledge the benefits of centralized control for their system, despite its challenges in managing complexity and scale due to technical and economic constraints. Centralized control designs are often dependent on the system structure and struggle to adapt to structural changes, requiring redesign from scratch if alterations are made. Additionally, they incur high costs and complexity in terms of online computation and control design.

3. Communications: implicit and explicit:

Cooperation is usually based on some forms of communication. Communication is a mode of interactions between multi-agent robots. With an efficient communication system, the robot is capable of interacting, sharing and exchanging information. Communication also determines the success in mobile robots cooperation. Based on research by Cao et al., there are

three types of communication structures which are (i) interaction via the environment, (ii) interaction via sensing, and (iii) interaction via communications.

Task	Communication devices/network	Interaction
Coordination of the ROBOCUP teams (middle size league) [3]	Robots equipped with communicating devices (off-the-shelf) radio modems and wireless Ethernet cards. Communication is based on underlying IP protocol either TCP-IP or UDP-IP	Agent to agent
Developing the control architectures for chain micro robots [5]	Command exchange protocol is used for communication between modules and PC by sending a message. The name of the protocol is <i>F³C</i> protocol	One to many agents (modules) for global. Agent to agents for local
The pusher robots work cooperatively to push the paralyzed robot to a specific point [6]	PC will send messages to the Mindstorm robot (paralyzed) to control Mirobot robots (pusher) by using infrared serial communications interface/transceiver	One way communication. One to all agents (broadcast) for global
Broadcast control framework for multi-agent coordination [22]	The broadcast signal sent from computer to all agents via Bluetooth	One way communication. One to all agents (broadcast)
Sign board based inter-robot communication in distributes robotic system [24]	Communication-based on conceptual mechanism of "sign-board" being used in inter-robot system	Agent to agent
Cooperative multi-robot system using Hello-Call Communication [25]	Each agent communicates together (chains) using "hello-call" protocol to extend their effective communication ranges	Agent to agent
Swarm robots control mobile robot using wireless sensor networks [36]	Using Wifi and three communication channels to interact between swarms for cooperation	One to many agents (broadcast)

4. Centralized and distributed control:

Centralized and distributed control mechanisms play pivotal roles in coordinating multi-agent systems, each presenting distinct advantages and challenges. However, it is imperative to delve into the problems and issues encountered in coordinating multi-agent systems using both control paradigms.

With centralized control, the global information of the environment is utilized to calculate paths, trajectories, or positions of agents in advance. This information can then be directly transmitted to the agents via suitable communication mediums, offering the advantage of direct access to centralized information.

On the other hand, distributed control has been demonstrated to be scalable, adaptive, flexible, and robust for multi-agent systems, operating effectively in both static and dynamic environments. Numerous researchers have successfully implemented distributed controllers for their multi-agent robot systems

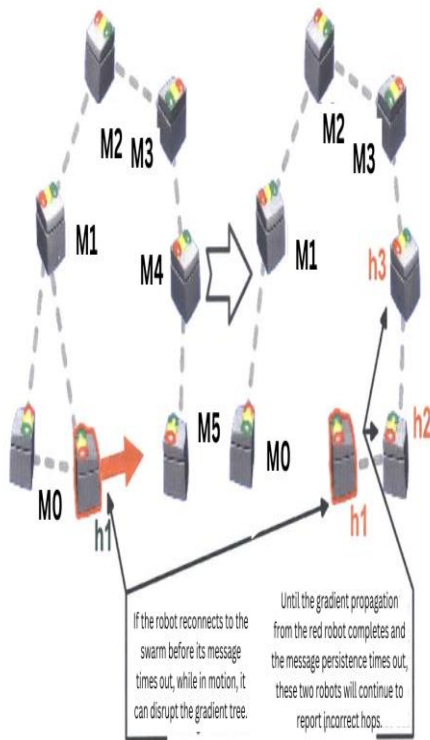


Fig.3.2- Centralized and distributed control

5. Formation:

Formation control stands as a crucial aspect in coordinating and managing a group of multi-agent robots. The fundamental objective is to enable robots to regulate their relative positions and orientations within the group to maneuver towards specific destinations. This area garners significant attention from researchers due to various factors including biological inspirations, intricate control challenges, and the burgeoning demand for multi-robot systems. Designing a controller for mobile robot formation entails addressing numerous issues such as formation stability, controllability of different formation patterns, safety considerations, and uncertainties within formations. The networking system in formation control poses additional challenges, as highlighted by Chen and Wang and Kiattisin. Communication delays and losses can destabilize formation control systems, necessitating robust communication protocols and network control systems.

To enhance realism in multi-agent robot coordination, effective communication system designs must be integrated with formation control, facilitating information exchange either locally or globally through sensing or wireless

networks. Additionally, exploring hybrid control frameworks for multi-agent robot formation control is emerging as a critical area for real-world applications.

6. Task Allocation:

The task allocation conundrum within multi-agent robotics has captured the interest of researchers, emphasizing the need for efficient assignment and execution of tasks by robots. Upon task assignment by the computer, it becomes imperative to devise suitable approaches to ensure successful allocation to the robots. Sarker et al. employed an attractive field model to facilitate self-organization of their robots while allocating tasks. Conversely, Tolmidis and Petrou advocated for multi-objective optimization in dynamic task allocation, yielding results showcasing scalability, a generic solution, and enhanced time and energy utilization. Additionally, Nagarajan and Thondiyath devised their own task allocation algorithm, demonstrating superior performance in minimizing turnaround time, makespan, and overall cost.

Task	Sensors
Multi-robot cooperatively collects pucks in the field	Robots equipped with pairs of photo sensors and pairs of IR sensors
Robot teams track the target and push the box cooperatively	Robots equipped with side sensors emitting different signals
Multi-robots work together to avoid other robots, remove obstacles, and pass objects	Real mobile robots equipped with CCD cameras
UAV allocates UGVs	UAV equipped with a video camera and onboard Inertial Measurement Unit (IMU); UGV equipped with onboard laser range finder sensor
Follower robots follow the leader while avoiding obstacles	Follower robots equipped with ultrasound sensors

7. Intelligence:

The intelligence exhibited by multi-agent robots in cooperative tasks hinges on the efficacy of their controllers, which significantly influence their performance. The evolution of Multi-Agent Robotic Systems (MARS) reflects a continuous increase in intelligence, mirroring technological advancements. With the advent of artificial intelligence (AI), researchers have increasingly turned to AI approaches in controller design.

Various AI methodologies have been employed by researchers in developing multi-agent controllers. Fuzzy logic and neural networks stand out as robust and effective approaches in multi-agent robot control design.

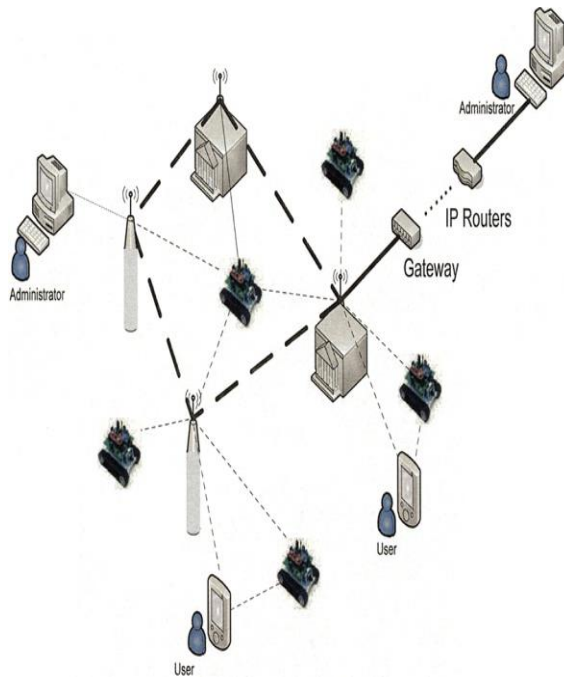


Fig.3.3- Multi-agent robot control design

IV. RESULTS AND DISCUSSIONS:

The implementation of the methodology for swarm robotics in multi-agent systems has unequivocally showcased the effectiveness of swarm intelligence principles. By harnessing decentralized, self-organized systems inspired by natural phenomena, such as social insects' collective behaviors, the implementation has underscored the inherent adaptability and versatility of swarm robotics. This approach has enabled multi-agent systems to tackle a diverse array of challenges in cooperative behavior among agents, ranging from task allocation and formation control to communication and intelligence integration. Through the successful deployment of various control architectures, including reactive, deliberative, and hybrid approaches, the implementation has demonstrated the ability to navigate dynamic environments while ensuring efficient coordination among agents. Furthermore, the utilization of both centralized and distributed control mechanisms has provided valuable insights into optimizing coordination strategies to suit different scenarios. Overall, the implementation has not only validated the efficacy of swarm intelligence principles but also highlighted the immense potential of swarm robotics in addressing complex real-world challenges through collaborative and adaptive behaviors among agents.

The methodology implemented in the study underscores the versatility and effectiveness of swarm robotics in addressing complex challenges inherent in multi-agent systems. By leveraging decentralized, self-organized systems inspired by natural swarm intelligence, the study demonstrates how individual robots can interact autonomously to achieve collective goals without the need for centralized control. This approach enables robots to adapt dynamically to changing environments and tasks, mirroring the collaborative behaviors observed in social insects like ants and bees. Furthermore, the study explores various control architectures, including reactive, deliberative, and hybrid approaches, each offering distinct advantages and challenges in coordinating multi-agent systems. While reactive architectures provide flexibility and robustness through decentralized control, deliberative architectures offer centralized coordination for enhanced efficiency. Additionally, the integration of communication systems and task allocation algorithms enhances the coordination and cooperation among agents, facilitating the successful execution of tasks.

V. CONCLUSION:

This paper presents a comprehensive review of cooperative multi-agent robotic systems (MARS), showcasing their potential to revolutionize various real-world applications through robust cooperation and coordination. A key takeaway from this research is the paramount importance of enhancing controller design and communication systems to achieve optimal performance among agents. As technology continues to evolve, there is immense potential for further advancements in MARS, with cooperative agents poised to make significant contributions across diverse domains. Future research endeavors could focus on refining controller designs and communication protocols, leveraging advancements in artificial intelligence and optimization techniques to enhance efficiency and performance. Additionally, addressing emerging challenges such as mobility, scalability, flexibility, global connectivity, and dynamic environments will be crucial for unlocking the full potential of MARS in the future. This paper lays the groundwork for future exploration in the field, offering valuable insights and recommendations for researchers seeking to expand the frontiers of MARS.

VI. REFERENCES:

- [1]Oliveira E, Fischer K, Stepankova O. Multi Agent Systems: Which Research for Which Applications. *Robotics and Autonomous Systems*. 1999;27:91-106
- [2]Parker, L.E. ALLIANCE: An architecture for fault-tolerant, cooperative control of heterogeneous mobile robots. *IEEE Transactions on Robotics and Automation*, 14, 220-240.
- [3]Asama, H., Matsumoto, A., & Ishida, Y. Design of an autonomous and distributed robot system: ACTRESS. In: *IEEE/RSJ International Workshop on Intelligent Robots and Systems* (pp. 283-290).
- [4]Candea, C., Hu, H., Iocchi, L., Nardi, D., & Piaggio, M. Coordination in multi-agent RoboCup teams. *Robotics and Autonomous Systems*, 36, 67-86.
- [5]Rosa, L., Cagnetti, M., Nicastrò, A., Alvarez, P., & Oriolo, G. Multi-task cooperative control in a heterogeneous ground-air robot team. In: *3rd IFAC Workshop on Multivehicle Systems* (pp. 53-58).
- [6]Hackwood S, Beni G. Self organization of sensors for swarm intelligence. In: *IEEE International Conference on Robotics and Automation*. pp. 819-829
- [7]Brunete, A., Hernando, M., Gambao, E., & Torres, J.E. A behavior-based control architecture for heterogeneous modular, multi-configurable, chained micro-robots. *Robotics and Autonomous Systems*, 60, 1607-1624.
- [8]Simonin, O., & Grunder, O. A cooperative multi-robot architecture for moving a paralyzed robot. *Mechatronics*, 19, 463-470.
- [9]Cao, Y., Fukunaga, A., & Kahng, A. Cooperative mobile robotics: Antecedents and directions. *Journal of Autonomous Robots*, 4, 1-23.
- [10]Yan, Z., Jouandeau, N., & Cherif, A.A. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotics Systems*, 10, 1-18.
- [11]Li J, Yan W. Consensus problems for multi agent system with broadcasting type topology. In: *2012 Second International Conference on Instrumentation & Measurement, Computer, Communication and Control*. pp. 967-970

SOURCE CODE:

HTML:

```
<!DOCTYPE html>

<html>

  <head>

    <meta http-equiv="Content-Security-Policy" content="default-src 'self' data: gap:
https://ssl.gstatic.com 'unsafe-eval'; style-src 'self' 'unsafe-inline'; media-src *">

    <meta name="format-detection" content="telephone=no">

    <meta name="msapplication-tap-highlight" content="no">

    <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1,
minimum-scale=1, width=device-width">

    <link rel="stylesheet" type="text/css" href="css/index.css">

    <title>Hello World</title>

  </head>

  <body>

    <div class="app">

      <h1>Apache Cordova</h1>

      <div id="deviceready" class="blink">

        <p class="event listening">Connecting to Device</p>

        <p class="event received">Device is Ready</p>

      </div>

    </div>

    <script type="text/javascript" src="cordova.js"></script>

    <script type="text/javascript" src="js/index.js"></script>

  </body>

</html>
```

Install.sql

```
SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";
CREATE TABLE `customers` (
  `c_id` int(11) NOT NULL,
  `c_name` varchar(256) NOT NULL,
```

```

`c_mobile` varchar(15) NOT NULL,
`c_email` varchar(100) NOT NULL,
`c_address` varchar(256) NOT NULL,
`c_created_date` datetime NOT NULL,
`c_pwd` varchar(100) DEFAULT NULL,
`c_isactive` varchar(11) NOT NULL DEFAULT '1',
`c_modified_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/* Table structure for table `drivers` */
CREATE TABLE `drivers` (
  `d_id` int(11) NOT NULL,
  `d_name` varchar(100) NOT NULL,
  `d_mobile` varchar(15) NOT NULL,
  `d_address` varchar(250) NOT NULL,
  `d_age` int(11) NOT NULL,
  `d_licenseno` varchar(100) NOT NULL,
  `d_license_expdate` date NOT NULL,
  `d_total_exp` int(11) NOT NULL,
  `d_doj` date NOT NULL,
  `d_ref` varchar(256) DEFAULT NULL,
  `d_is_active` int(11) NOT NULL DEFAULT '1',
  `d_created_by` varchar(100) NOT NULL,
  `d_created_date` datetime NOT NULL,
  `d_modified_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Table structure for table `email_template`*/
CREATE TABLE `email_template` (
  `et_id` int(11) NOT NULL,
  `et_name` varchar(256) NOT NULL,
  `et_subject` varchar(100) NOT NULL,
  `et_body` longtext NOT NULL,
  `et_created_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```
INSERT INTO `email_template` (`et_id`, `et_name`, `et_subject`, `et_body`, `et_created_date`)
VALUES
```

```
(1, 'booking', 'Booking Confirmation - VMS', '<p>Dear Customer,<p>\r\n\r\n<p>Thank you for choosing
VMS<p>\r\n\r\n<p>We look forward to welcoming you to strat
trip.<p>\r\n\r\n<p>{{bookingdetails}}<p>\r\n\r\n<p>Our professional and friendly staff are committed to
ensuring your travel is both enjoyable and comfortable.<p>\r\n\r\n<p>Should you have any requests
prior to your travel, please do not hesitate to contact us and we will endeavor to assist you whenever
possible.<p>', '2020-07-30 19:47:12'),
```

```
/*Table structure for table `fuel`*/
```

```
CREATE TABLE `fuel` (
  `v_fuel_id` int(10) NOT NULL,
  `v_id` int(100) NOT NULL,
  `v_fuel_quantity` varchar(255) CHARACTER SET latin1 NOT NULL,
  `v_odometerreading` varchar(100) DEFAULT NULL,
  `v_fuelprice` varchar(255) CHARACTER SET latin1 NOT NULL,
  `v_fuelfilldate` date NOT NULL,
  `v_fueladdedby` varchar(100) NOT NULL,
  `v_fuelcomments` varchar(256) NOT NULL,
  `v_created_date` datetime NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
/* Table structure for table `geofences`*/
```

```
CREATE TABLE `geofences` (
  `geo_id` int(11) NOT NULL,
  `geo_name` varchar(128) NOT NULL,
  `geo_description` varchar(128) DEFAULT NULL,
  `geo_area` varchar(4096) NOT NULL,
  `geo_vehicles` varchar(256) NOT NULL,
  `geo_createddate` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `geo_modifieddate` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE
CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
/*Table structure for table `geofence_events`*/
```

```
CREATE TABLE `geofence_events` (
  `ge_id` int(11) NOT NULL,
  `ge_v_id` varchar(11) NOT NULL,
  `ge_geo_id` varchar(11) NOT NULL,
  `ge_event` varchar(256) NOT NULL,
```

```

`ge_timestamp` varchar(100) NOT NULL,
`ge_created_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/* Table structure for table `incomeexpense` */
CREATE TABLE `incomeexpense` (
  `ie_id` int(11) NOT NULL,
  `ie_v_id` varchar(100) NOT NULL,
  `ie_date` date NOT NULL,
  `ie_type` varchar(100) NOT NULL,
  `ie_description` varchar(256) NOT NULL,
  `ie_amount` int(100) NOT NULL,
  `ie_created_date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `ie_modified_date` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/* Table structure for table `login` */
CREATE TABLE `login` (
  `u_id` int(11) NOT NULL,
  `u_name` varchar(250) NOT NULL,
  `u_username` varchar(250) NOT NULL,
  `u_password` varchar(250) NOT NULL,
  `u_isactive` varchar(100) NOT NULL DEFAULT '1',
  `u_email` varchar(256) NOT NULL,
  `u_created_date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

MAIN.CSS

```

@font-face {
  font-family: 'FontAwesome';
  src: url('../font/fontawesome-webfont.eot?v=3.2.1');
  src: url('../font/fontawesome-webfont.eot?#iefix&v=3.2.1') format('embedded-opentype'),
url('../font/fontawesome-webfont.woff?v=3.2.1') format('woff'), url('../font/fontawesome-
webfont.ttf?v=3.2.1') format('truetype'), url('../font/fontawesome-
webfont.svg#fontawesomeregular?v=3.2.1') format('svg');
  font-weight: normal;
  font-style: normal;

```

```

}
[class^="icon-"],
[class*=" icon-"] {
    font-family: FontAwesome;
    font-weight: normal;
    font-style: normal;
    text-decoration: inherit;
    -webkit-font-smoothing: antialiased;
    *margin-right: .3em;
}
[class^="icon-"]:before,
[class*=" icon-"]:before {
    text-decoration: inherit;
    display: inline-block;
    speak: none;
}
.icon-large:before {
    vertical-align: -10%;
    font-size: 1.3333333333333333em;
}
.icon-5x.icon-border {
    border-width: 5px;
    -webkit-border-radius: 7px;
    -moz-border-radius: 7px;
    border-radius: 7px;
}
.pull-right {
    float: right;
}
.pull-left {
    float: left;
}
[class^="icon-"].pull-left,
[class*=" icon-"].pull-left {

```

```

    margin-right: .3em;
}
[class^="icon-"].pull-right,
[class*=" icon-"].pull-right {
    margin-left: .3em;
}
.icon-white,
.nav-pills>.active>a[class^="icon-"],
.nav-pills>.active>a[class*=" icon-"],
.nav-list>.active>a[class^="icon-"],
.nav-list>.active>a[class*=" icon-"],
.navbar-inverse .nav>.active>a[class^="icon-"],
.navbar-inverse .nav>.active>a[class*=" icon-"],
.dropdown-menu>li>a:hover>[class^="icon-"],
.dropdown-menu>li>a:hover>[class*=" icon-"],
.dropdown-menu>.active>a[class^="icon-"],
.dropdown-menu>.active>a[class*=" icon-"],
.dropdown-submenu:hover>a[class^="icon-"],
.dropdown-submenu:hover>a[class*=" icon-"] {
    background-image: none;
}
.btn [class^="icon-"].icon-large,
.nav [class^="icon-"].icon-large,
.btn [class*=" icon-"].icon-large,
.nav [class*=" icon-"].icon-large {
    line-height: .9em;
}
.btn [class^="icon-"].icon-spin,
.nav [class^="icon-"].icon-spin,
.btn [class*=" icon-"].icon-spin,
.nav [class*=" icon-"].icon-spin {
    display: inline-block;
}
.nav-tabs [class^="icon-"],

```

```

.nav-pills [class^="icon-"],
.nav-tabs [class*=" icon-"],
.nav-pills [class*=" icon-"],
.nav-tabs [class^="icon-"].icon-large,
.nav-pills [class^="icon-"].icon-large,
.nav-tabs [class*=" icon-"].icon-large,
.nav-pills [class*=" icon-"].icon-large {
    line-height: .9em;
}

.btn [class^="icon-"].pull-left.icon-2x,
.btn [class*=" icon-"].pull-left.icon-2x,
.btn [class^="icon-"].pull-right.icon-2x,
.btn [class*=" icon-"].pull-right.icon-2x {
    margin-top: .18em;
}

.btn [class^="icon-"].icon-spin.icon-large,
.btn [class*=" icon-"].icon-spin.icon-large {
    line-height: .8em;
}

.btn.btn-small [class^="icon-"].pull-left.icon-2x,
.btn.btn-small [class*=" icon-"].pull-left.icon-2x,
.btn.btn-small [class^="icon-"].pull-right.icon-2x,
.btn.btn-small [class*=" icon-"].pull-right.icon-2x {
    margin-top: .25em;
}

.btn.btn-large [class^="icon-"],
.btn.btn-large [class*=" icon-"] {
    margin-top: 0;
}

.btn.btn-large [class^="icon-"].pull-left.icon-2x,
.btn.btn-large [class*=" icon-"].pull-left.icon-2x,
.btn.btn-large [class^="icon-"].pull-right.icon-2x,
.btn.btn-large [class*=" icon-"].pull-right.icon-2x {

```

```

    margin-top: .05em;
}
.btn.btn-large [class^="icon-"].pull-left.icon-2x,
.btn.btn-large [class*=" icon-"].pull-left.icon-2x {
    margin-right: .2em;
}
.btn.btn-large [class^="icon-"].pull-right.icon-2x,
.btn.btn-large [class*=" icon-"].pull-right.icon-2x {
    margin-left: .2em;
}
.nav-list [class^="icon-"],
.nav-list [class*=" icon-"] {
    line-height: inherit;
}
.icon-stack {
    position: relative;
    display: inline-block;
    width: 2em;
    height: 2em;
    line-height: 2em;
    vertical-align: -35%;
}
.icon-stack [class^="icon-"],
.icon-stack [class*=" icon-"] {
    display: block;
    text-align: center;
    position: absolute;
    width: 100%;
    height: 100%;
    font-size: 1em;
    line-height: inherit;
    *line-height: 2em;
}

```

Bootstrap.js

```
(function (global, factory) {
  typeof exports === 'object' && typeof module !== 'undefined' ? factory(exports, require('jquery')) :
  typeof define === 'function' && define.amd ? define(['exports', 'jquery'], factory) :
  (factory((global.bootstrap = {}), global.jQuery));
}(this, (function (exports, $) { 'use strict';
$ = $ && $.hasOwnProperty('default') ? $['default'] : $;
function _defineProperties(target, props) {
  for (var i = 0; i < props.length; i++) {
    var descriptor = props[i];
    descriptor.enumerable = descriptor.enumerable || false;
    descriptor.configurable = true;
    if ("value" in descriptor) descriptor.writable = true;
    Object.defineProperty(target, descriptor.key, descriptor);
  }
}
function _createClass(Constructor, protoProps, staticProps) {
  if (protoProps) _defineProperties(Constructor.prototype, protoProps);
  if (staticProps) _defineProperties(Constructor, staticProps);
  return Constructor;
}
function _extends() {
  _extends = Object.assign || function (target) {
    for (var i = 1; i < arguments.length; i++) {
      var source = arguments[i];
      for (var key in source) {
        if (Object.prototype.hasOwnProperty.call(source, key)) {
          target[key] = source[key];
        }
      }
    }
  };
  return _extends.apply(this, arguments);
}
```

```

}
function _inheritsLoose(subClass, superClass) {
  subClass.prototype = Object.create(superClass.prototype);
  subClass.prototype.constructor = subClass;
  subClass.__proto__ = superClass;
}

var transition = false;
var MAX_UID = 1000000; // Shoutout AngusCroll (https://goo.gl/pxwQGp)
function toType(obj) {
  return {}.toString.call(obj).match(/\s([a-zA-Z]+)/)[1].toLowerCase();
}
function getSpecialTransitionEndEvent() {
  return {
    bindType: transition.end,
    delegateType: transition.end,
    handle: function handle(event) {
      if ($$$1(event.target).is(this)) {
        return event.handleObj.handler.apply(this, arguments); // eslint-disable-line prefer-rest-params
      }
      return undefined; // eslint-disable-line no-undefined
    }
  };
}
function transitionEndTest() {
  if (typeof window !== 'undefined' && window.QUnit) {
    return false;
  }
  return {
    end: 'transitionend'
  };
}
function transitionEndEmulator(duration) {
  var _this = this;
  var called = false;

```

```

$$$1(this).one(Util.TRANSITION_END, function () {
    called = true;
});
setTimeout(function () {
    if (!called) {
        Util.triggerTransitionEnd(_this);
    }
    return this;
}

function escapeld(selector) {
    selector = typeof $$$1.escapeSelector === 'function' ? $$$1.escapeSelector(selector).substr(1) :
    selector.replace(/(:|\.|\\|\\]|,|=|@)/g, '\\$1');
    return selector;
}

var Util = {
    TRANSITION_END: 'bsTransitionEnd',
    getUID: function getUID(prefix) {
    supportsTransitionEnd: function supportsTransitionEnd() {
        return Boolean(transition);
    },
    isElement: function isElement(obj) {
        return (obj[0] || obj).nodeType;
    },
    typeCheckConfig: function typeCheckConfig(componentName, config, configTypes) {
        for (var property in configTypes) {
            if (Object.prototype.hasOwnProperty.call(configTypes, property)) {
                var expectedTypes = configTypes[property];
                var value = config[property];
                var valueType = value && Util.isElement(value) ? 'element' : toType(value);
            }
        }
    };
    setTransitionEndSupport();
    return Util;
};

```