



VAN-D : BREAKDOWN SYSTEM

MINI PROJECT -I REPORT

Submitted by

NIVAS P (21EPCI036)

DHANASEELAN S (21EPCI009)

VAISHAK CJ (21EPCI054)

in partial fulfillment for the award of the degree of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(5 Year Integrated)

SRI KRISHNA COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University Chennai - 600 025)

MARCH 2024



BONAFIDE CERTIFICATE

Certified that this project report “**VAN-D : BREAKDOWN SYSTEM**” is the bonafide work of “**NIVAS P (21EPCI036), DHANASEELAN S (21EPCI009), VAISHAK CJ (21EPCI054)**” who carried out the project work under my supervision.

SIGNATURE

HEAD OF THE DEPARTMENT

Department of M.Tech. CSE
Sri Krishna College of Engg. & Tech.,
Coimbatore – 641 008

SIGNATURE

Ms.T.Dureen V Rayen

SUPERVISOR

Assistant Professor
Department of M.Tech. CSE
Sri Krishna College of Engg. & Tech.,
Coimbatore – 641 008

Submitted for the Mini Project viva-voce examination held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

At this juncture, we take the opportunity to convey our sincere thanks and gratitude to the management of the college for providing all the facilities to us. We wish to convey our gratitude to our college principal, **Dr.S.Sophia**, for forwarding us to do our project and offering adequate duration to complete our project.

We would like to express our grateful thanks to Professor, Head of the Department, Department of M. Tech. Computer Science and Engineering for her encouragement and valuable guidance to this project.

We extend our gratitude to our Project Coordinator **Mr. Sreeraj S**, Assistant Professor, Department of M. Tech Computer Science and Engineering for his constant support and immense help at all stages of the project.

We are grateful to our project guide **Ms.T.Dureen V Rayen**, Associate Professor, Department of M. Tech. Computer Science and Engineering for her contributions of time to our project. We are also grateful for the internal and external members of the project viva voice.

ABSTRACT

The Vehicle Assistance and Defect fixer system (VAND) or Vehicle Breakdown Management System (VBMS) stands as a comprehensive and indispensable tool for projects managing a fleet of vehicles, such as those in logistics, transportation, and construction. At its core, VBMS incorporates a sophisticated mechanic search functionality, allowing users to promptly access a list of trusted mechanics at their location or nearby. The inclusion of only authorized and reliable mechanics in the platform's database ensures a high standard of service. Harnessing the power of location-based services, the system not only accurately identifies the user's location but also provides a visual map interface illustrating the available mechanics in the vicinity, facilitating informed decision-making. The system extends its support with a range of roadside assistance services tailored for cars, covering diverse issues from towing to minor repairs. Real-time communication tools, including chat and call features, establish a direct link between users and mechanics, fostering efficient and timely problem resolution. In essence, VAND or VBMS emerges as a versatile and indispensable ally in navigating the unexpected challenges posed by vehicle breakdowns, ensuring a seamless and efficient breakdown management process for projects relying on a fleet of vehicles.

TABLE OF CONTENT

CHAPTER NO	TITLE	PG NO
	ABSTRACT	iii
	LIST OF FIGURES	vi
	LIST OF ABBREVIATIONS	vii
1	INTRODUCTION	1
2	LITERATURE SURVEY 2.1 Vehicle Breakdown Assistance 2.2 Multi-Agent Reinforcement Learning 2.3 Recent Advances in Connected Vehicle Technologies	6 6 7 8
3	SYSTEM ANALYSIS 3.1 Existing System 3.2 Proposed System	10 10 12
4	SYSTEM SPECIFICATION	15

5	PROJECT DESCRIPTION	17
	5.1 Customer function	17
	5.2 Mechanic function	18
	5.3 Admin function	20
	5.4 Integration with Map	21
	5.5 Testing And Evaluation	22
	5.6 Advantages of VAN-D Breakdown System	22
	5.7 Challenges of VAN-D Breakdown System	23
	5.8 Tools And Technologies Used	24
6	SOURCE CODE	27
7	SCREENSHOTS	46
8	CONCLUSIONS	49
	8.1 Conclusion	
	8.2 Future work	
	REFERENCES	50

LIST OF FIGURES

Figure No	Name	Page No
5.1.1	Dataflow Diagram(Customer)	24
5.2.1	Dataflow Diagram(Machanic)	25
7.1	Home Page	39
7.2	Customer Dashboard	39
7.3	Admin Dashboard	40
7.4	Machanic Dashboard	40
7.5	Invoice Page	41
7.6	Payment Page	41

LIST OF ABBREVIATIONS

ABBREVIATIONS

EXPANSIONS

CVT	Connected Vehicle Technology
HTTP	Hypertext Transfer Protocol
MARL	Multi-Agent Reinforcement Learning
QR	Quick Response Code
UPI	Unified Payments Interface
VAND	Vehicle Assistance and Defect Fixer System
VBMS	Vehicle Breakdown Management System
V2X	Vehicle-to-Everything

CHAPTER I

INTRODUCTION

The introduction strategically captivates the reader by immersing them in the intricate and challenging realm of vehicular breakdowns. It artfully weaves a narrative that vividly portrays the complexities associated with managing such incidents, elucidating the myriad hurdles and intricacies that can unfold in unexpected situations on the road. By acknowledging the inherent difficulties and uncertainties of vehicular breakdowns, the introduction effectively communicates the urgent need for a sophisticated solution that possesses the finesse to navigate and resolve these challenges seamlessly.

Taking the narrative a step further, the introduction skillfully transitions to spotlight the broader goals of the project. It elevates the reader's understanding beyond the immediate issues, providing a panoramic view of the initiative's scope and aspirations. This holistic perspective imparts insight into the project's ambitious aims, revealing that it strives not merely to address the surface-level problems of breakdown incidents but to revolutionize the entire landscape of incident management within the automotive domain. The introduction sets the stage for a comprehensive solution that extends far beyond quick fixes, promising a transformative approach to addressing the complexities of vehicular breakdowns.

Furthermore, the introduction deftly underscores the transformative potential embedded in the project. It articulates a vision of innovation and positive change, positioning the proposed solution not as a mere reactive measure but as a dynamic catalyst for a paradigm shift in incident management. It sparks curiosity about how the proposed solution will not

only address current challenges but also lay the foundation for a more efficient, proactive, and advanced approach to vehicular breakdown management.

Inner Components:

User Registration:

The User Registration component is the initial step where individuals provide essential details to create an account within the system. This information may include personal details, vehicle information, and contact information. The registration process establishes a personalized connection between the user and the system, enabling access to the full range of services.

User Login:

Following registration, the User Login component ensures secure access for authorized users. Users are required to log in to the system using their credentials, providing a layer of security. This step protects user data and ensures that only authenticated individuals can utilize the breakdown assistance services.

Mechanic Database:

The Mechanic Database component maintains a comprehensive repository of trusted mechanics who have undergone a thorough vetting process. This may include background checks, certifications, and customer feedback analysis. Users can rely on this database to access skilled and reliable professionals when seeking breakdown assistance.

Request Assistance:

The Request Assistance component allows users to communicate their breakdown situation to the system. Upon receiving a request, the system identifies the nearest available mechanic based on the user's location and dispatches them promptly.

Real-time Tracking:

The Real-time Tracking component provides users with the ability to monitor the location and estimated time of arrival of the dispatched mechanic. This feature adds transparency to the breakdown assistance process, allowing users to track the progress and plan accordingly.

Feedback and Ratings:

The Feedback and Ratings component encourages users to share their experiences and rate the service received from the mechanic. This valuable information contributes to the system's ongoing evaluation of mechanics, fostering a continuous improvement cycle and maintaining a high standard of service quality.

Payment Integration:

The Payment Integration component streamlines financial transactions within the system. Users can make secure and hassle-free payments directly through the application, providing a convenient and cashless solution.

Mechanic Verification System:

The Mechanic Verification System component ensures that only qualified and trustworthy professionals become part of the platform. Background checks, certifications, and analysis of customer feedback contribute to the vetting process, enhancing the reliability of the mechanics available in the system.

Emergency Contacts and Notifications:

Users can designate Emergency Contacts within the system. In the event of a breakdown, the system sends instant notifications to these contacts, keeping them informed about the user's situation and the ongoing breakdown assistance process..

Service History:

The Service History component maintains a detailed record of each user's past breakdown incidents, mechanics hired, and the feedback provided. This historical data serves multiple purposes, including personalized user experience, targeted service recommendations, and assisting mechanics in understanding the vehicle's maintenance needs.

Collaboration with Roadside Services:

The Collaboration with Roadside Services component broadens the scope of assistance by integrating with towing companies, roadside services, and emergency response units.

In summary, the extended explanation of the inner components reveals a multifaceted and intricately designed Vehicle Breakdown Assistance System. The inclusion of features such as real-time tracking, service history, payment integration, mechanic verification, and emergency contacts contributes to a holistic and user-centric approach, addressing the myriad challenges associated with vehicular breakdowns. The system not only provides immediate assistance but also focuses on long-term user satisfaction and system optimization. the introduction masterfully combines a nuanced acknowledgment of challenges with a clear articulation of ambitious goals and transformative potential.

CHAPTER 2

LITERATURE SURVEY

2.1 Vehicle Breakdown Assistance

This application is a crucial resource for individuals encountering vehicle breakdowns in remote areas, providing a streamlined solution to locate nearby mechanics promptly. By leveraging geolocation, users can access a curated list of approved mechanics, ensuring quick assistance. The system diligently monitors these mechanics to prevent any additional service fees, maintaining transparency and fairness. User feedback plays a pivotal role, enabling customers to share their experiences and aiding the admin in monitoring and upholding service standards. Access to the application is exclusive to registered users, ensuring a secure and personalized experience. This approach not only saves time but also enhances user safety and convenience. Moreover, the app facilitates seamless payment processing for vehicle repairs at reasonable rates, providing users with a hassle-free experience. The admin dashboard empowers administrators to oversee mechanic performance, review feedback, and manage approvals, ensuring the application's integrity and user satisfaction. In summary, this application is a lifeline for those navigating vehicle breakdowns in remote areas, offering a reliable, efficient, and user-centric solution to secure prompt and trustworthy mechanical assistance.

2.2 Multi-Agent Reinforcement Learning for Dynamic Dispatching of Roadside Assistance Vehicles

This paper presents a comprehensive approach to optimizing the dispatch of roadside assistance vehicles in real-time, focusing on the development of a multi-agent reinforcement learning (MARL) algorithm. The algorithm is designed to consider a wide range of factors that impact dispatch decisions, including the location of breakdowns, the availability of resources such as tow trucks and mechanics, current traffic conditions, and various priority criteria such as the severity of breakdowns and the type of vehicles involved.

The MARL algorithm works by deploying a group of intelligent agents, each representing a different aspect of the dispatch process. These agents learn and adapt their strategies through interactions with the environment, receiving feedback based on the outcomes of their decisions. This adaptive learning process allows the agents to continuously improve their dispatch decisions, leading to more efficient resource allocation and reduced response times.

The algorithm's effectiveness is demonstrated through simulations and real-world experiments, showing significant improvements in key metrics such as response time, resource utilization, and overall efficiency. The paper concludes with a discussion of the potential applications of the MARL algorithm in other transportation and logistics scenarios, highlighting its versatility and potential for widespread adoption in related fields.

2.3 Recent Advances in Connected Vehicle Technologies for Roadside Assistance Systems

This paper critically examines the profound impact of Connected Vehicle Technology (CVT) on the enhancement of roadside assistance systems. By concentrating on key elements such as real-time vehicle diagnostics, automatic breakdown detection, and the efficient dispatch of assistance vehicles, the paper explores the manifold ways in which CVT can revolutionize the landscape of breakdown assistance services. The first highlighted aspect is the realm of real-time vehicle diagnostics. Leveraging CVT, this technology enables the continuous monitoring of a vehicle's health and performance by assimilating data from various onboard sensors. This proactive approach empowers the roadside assistance system to identify potential issues before they escalate into breakdowns, significantly improving the predictability and prevention of vehicular issues.

In tandem, the paper underscores the significance of automatic breakdown detection facilitated by CVT. Through seamless communication between the vehicle and roadside assistance infrastructure, CVT enables the swift identification of anomalies or signs of impending breakdowns. This early detection mechanism ensures that assistance processes can be initiated promptly, minimizing the inconvenience experienced by the driver.

The efficiency of roadside assistance systems is further heightened through the utilization of CVT for the dispatch of assistance vehicles. With real-time location data provided by CVT, assistance vehicles can be dispatched precisely based on factors such as proximity, expertise, and availability. This targeted dispatching ensures a rapid and effective response to breakdown incidents, optimizing the overall assistance workflow.

A pivotal application of CVT in breakdown assistance, as expounded in the paper, is Vehicle-to-Everything (V2X) communication. This interconnected network allows vehicles to exchange critical data with roadside infrastructure. By sharing information on the vehicle's health and status, this communication enables the infrastructure to detect breakdowns promptly. Moreover, V2X facilitates real-time updates on traffic conditions and the availability of assistance resources, contributing to a more informed and efficient response mechanism.

The paper accentuates CVT's role in enhanced data sharing for traffic updates. Through V2X communication, vehicles can share pertinent information about breakdowns, road closures, and traffic conditions with the roadside infrastructure.

CHAPTER 3

SYSTEM ANALYSIS

3.1 Existing System:

The On Road Vehicle Breakdown Assistance system project is a comprehensive solution designed to assist users in finding trusted mechanics quickly and efficiently in the event of a vehicle breakdown. The system allows users to search for mechanics at any location or nearby areas, ensuring timely assistance during unexpected mechanical issues. Only mechanics who have been vetted and approved are listed, providing users with reliable service options. The system focuses on providing road assistance for cars, offering services such as jump-starts, tire changes, and towing to repair facilities. Additionally, the system includes mechanics capable of performing on-site repairs, eliminating the need for towing in some cases. After receiving assistance, users are encouraged to provide feedback, helping to maintain the quality of service. Overall, the project aims to provide users with a convenient and trustworthy solution for their roadside assistance needs.

Customer Module:

Register: Users must register with the application by providing their basic details, such as name, contact information, and vehicle details. This registration process ensures that only legitimate users can access the service and helps in personalizing the user experience.

Login: Registered users can log in using their credentials. This step is necessary to access the full range of services provided by the application, including viewing mechanics and requesting assistance.

View Details: After logging in, users can view a list of approved mechanics in their area. This list includes details such as mechanic name, location, contact information, and user ratings. Users can use this information to make informed decisions when selecting a mechanic.

Search Records & Call: Users can search for mechanics based on their location and the type of service needed. The search functionality helps users find mechanics quickly and efficiently. Once a mechanic is selected, users can call them directly from the application, streamlining the process of requesting assistance.

Post Feedback: After receiving service, users are required to provide feedback on their experience. This feedback is crucial as it helps maintain the quality of service and informs other users about the reliability of the mechanics listed in the application. Users can rate the mechanic's service and provide comments to further elaborate on their experience.

Admin Module:

Customers: The admin can view, delete, and edit customer details. This helps manage user accounts and ensure that only authorized users access the application. The admin can also use this information to communicate with users if needed.

Mechanics: The admin can view, delete, and edit mechanic details. This includes verifying new mechanics and managing their profiles. The admin can also assign specific tasks or responsibilities to mechanics based on their expertise and availability.

View All Feedbacks: The admin can view feedback from users about the services provided by mechanics. This feedback helps in monitoring the performance of mechanics and ensuring customer satisfaction. The admin can take appropriate actions based on the feedback received, such as providing additional training to mechanics or removing them from the list of approved mechanics if necessary

3.2 Proposed System:

The Mechanic Module enhances the existing system by offering mechanics a platform to directly manage their profiles and services, reducing the burden on the admin and improving efficiency. Here's a more detailed explanation:

Registration and Approval: Mechanics register by providing essential details. The admin reviews and approves these registrations, ensuring that only verified mechanics are listed.

Profile Management: Once approved, mechanics can log in and manage their profiles. They can update their contact information, service details, working hours, and any other relevant information.

Service Availability: Mechanics can specify the types of services they offer, such as roadside assistance, repairs, maintenance, etc. This information helps users find mechanics based on their specific needs.

Feedback and Improvement: Mechanics can view feedback provided by users/customers. This feedback is crucial for improving their services and maintaining customer satisfaction. Mechanics can use this feedback to address any issues and enhance their service quality.

Communication: The module allows for direct communication between mechanics and users. Mechanics can respond to service requests, provide updates on repair progress, and address any concerns users may have.

Transparency and Trust: By allowing mechanics to manage their profiles and interact directly with users, the module promotes transparency and trust. Users can make informed decisions based on mechanic profiles and feedback from other users.

Customer Module:

The Customer Module can be further enhanced by integrating a map function, which provides users with visual information about the location of mechanics. Here's how the map function can be implemented:

Location Identification: Mechanics can specify their exact location or provide an address when registering. This information is used to plot their location on the map.

Map Display: Users can view a map interface within the application, displaying the locations of mechanics in their area. Mechanics' profiles can be represented as pins on the map, making it easy for users to identify nearby options.

Search and Filter: Users can search for mechanics based on their location and filter results based on distance, services offered, ratings, and other criteria. The map updates dynamically to display relevant results based on the user's search parameters.

Interactive Features: The map interface allows users to interact with mechanic pins, providing additional information such as contact details, service availability, and user ratings when clicked.

Route Planning: Users can utilize the map to plan routes to the selected mechanic's location. Integration with navigation services allows users to get directions to the mechanic with ease.

Real-Time Updates: The map function can include real-time updates on mechanic availability and service status. Mechanics can update their status (e.g., available, busy, offline) to ensure users have accurate information.

Payment Option: Customer can pay using QR code by UPI transactions, any App can be used Phonepay, Googlepay, PayTM etc.

By integrating a map function, the Customer Module enhances user experience by providing visual context, facilitating easier decision-making, and enabling efficient route planning for users seeking roadside assistance.

Overall, the Mechanic Module streamlines the process of managing mechanic data, improves communication between mechanics and users, and enhances the overall user experience.

CHAPTER 4

SYSTEM SPECIFICATION

Software Specification

Server Side: Python 3.7.4(64-bit)

Python is a general-purpose interpreted, interactive, object oriented, and high-level programming language. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

Client Side: HTML, CSS, Bootstrap

Bootstrap provides a collection of CSS, JavaScript, and HTML templates and components that help developers create consistent and visually appealing user interfaces. It includes a grid system for laying out content, typography styles, form controls, navigation menus, and other UI elements.

IDE: Django

The software specification for the Django project includes both functional and non-functional requirements. Functionally, the application should support user authentication, allowing users to register, log in, and update their profiles. It should also enable mechanics to register, log in, manage their services, and receive feedback from users. Users should be able to search for mechanics based on location, services offered, and ratings, and provide feedback and ratings for mechanics. Finally, the codebase should be well-organized and documented for ease of maintenance and updates. The technology stack includes Django for backend development, HTML/CSS/JavaScript for frontend development, a relational database for

data storage, and a deployment platform for deployment. Additional features such as map integration and payment integration can enhance the application's functionality.

Back end: SQLite3

SQLite3 is a popular choice for developers seeking a lightweight, serverless, and self-contained relational database management system (RDBMS). Its ease of use and minimal configuration make it ideal for applications requiring a local or embedded database solution. SQLite3 databases are stored in a single file, simplifying deployment and management. Despite its lightweight nature, SQLite3 is ACID-compliant, ensuring data integrity and reliability. It supports a subset of SQL, making it compatible with standard SQL syntax and commands. While SQLite3 is suitable for small to medium-sized applications.

Server: XAMPP Server

This means you can simply download and install a single program and follow a few easy prompts to get your web development server up and running in the quickest time with the minimum hassle.

CHAPTER 5

PROJECT DESCRIPTION

There are several processes in this System it has various modules and Functions. These processes are:

5.1 CUSTOMER FUNCTION:

In the customer module of the system, customers can sign up and log in to access a range of features. They can initiate service requests for their vehicles by providing detailed information such as vehicle number, model, and a description of the problem. Once a request is approved by the admin, customers can view the cost estimate and the current status of the service. They can also delete pending requests if they change their mind or if the request has not been approved yet.

Customers have the ability to track the status of their requests, which can be in various stages such as Pending, Approved, Repairing, Repairing Done, or Released. They can view detailed invoices and repair information for their vehicles. Additionally, customers can provide feedback to the admin, helping to improve the overall service quality.

Profile management is also available, allowing customers to view and edit their profile details, including name, contact information, and vehicle details. They can also change their password and update other account settings as needed. These features provide customers with a user-friendly interface to request, track, and manage vehicle services efficiently.

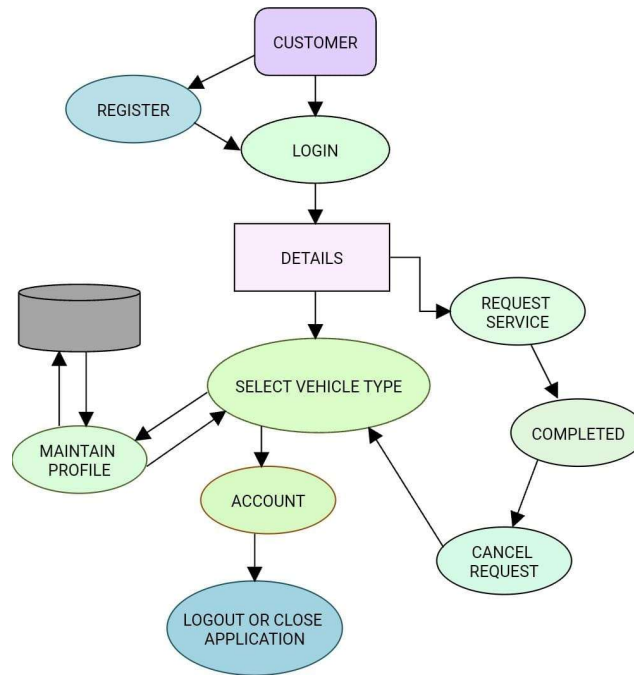


Fig 5.1.1 Dataflow

5.2 MECHANIC FUNCTION:

In the mechanic module of the system, mechanics can apply for a job by providing details such as skills, address, and mobile number. The admin will then hire (approve) mechanic accounts based on their skills and other criteria. Once approved, mechanics can log in to the system.

Mechanics can view the number of work orders (vehicles to repair) assigned to them. They can change the status of the service for each work order, indicating whether it is 'Repairing' or 'Repairing Done' based on the progress of the work.

Mechanics can also view their salary details and track how many vehicles they have repaired so far. They can send feedback to the admin about the job or the system, helping to improve the overall service quality.

Profile management is available for mechanics to view and edit their profile details such as name, contact information, and skills. They can also change their password and update other account settings as needed. These features provide mechanics with a user-friendly interface to manage their work and track their progress efficiently.

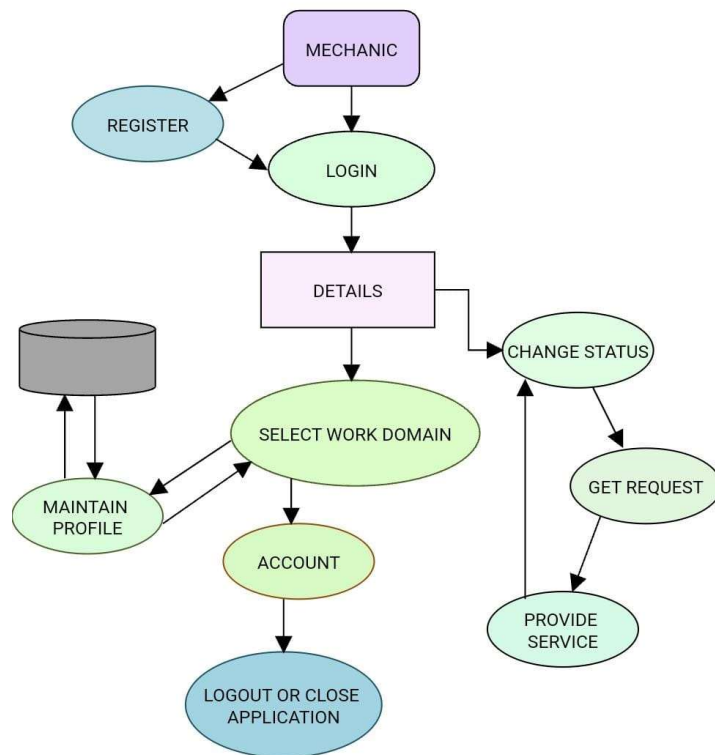
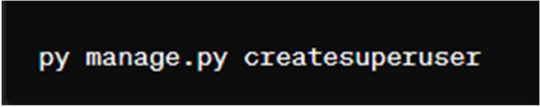


Fig 5.2.1 Dataflow

These features provide mechanics with a user-friendly interface to manage their work and track their progress efficiently. They can also change their password and update other account settings as needed. They can send feedback to the admin about the job or the system, helping to improve the overall service quality.

5.3 ADMIN FUNCTIONS:

To begin, the admin needs to create a superuser account by running the ``createsuperuser`` command in the command prompt and providing a username, email, and password. Once logged in, the admin dashboard displays key metrics such as the number of customers, mechanics, and recent service orders.



```
py manage.py createsuperuser
```

The admin has the ability to manage customers by viewing, adding, updating, or deleting customer records. Invoices for customers display the total sum of all requests made by the same customer. Similarly, the admin can manage mechanics by viewing, adding, updating, or deleting mechanic records. Mechanics can be approved based on their skills and qualifications, and their salaries can be set and updated by the admin.

For service requests, the admin can view, add, update, or delete service requests made by customers. The admin can also make service requests on behalf of customers. Service requests can be approved by the admin, who can then assign them to mechanics for repair and provide cost estimates based on problem descriptions. The admin can view the costs associated with all service requests, both approved and pending.

Additionally, the admin can access feedback provided by both customers and mechanics. This comprehensive set of features empowers the admin to efficiently manage the platform, oversee service operations, and ensure customer satisfaction.

5.4 INTEGRATION WITH MAP:

The map feature in the project enhances the user experience by providing a visual representation of mechanic locations. Mechanics can specify their exact location or provide an address when registering, which is then plotted on the map. Users can view a map interface within the application, displaying the locations of mechanics in their area as pins on the map. This visual representation makes it easy for users to identify nearby mechanics.

Users can search for mechanics based on their location and filter results based on distance, services offered, ratings, and other criteria. The map dynamically updates to display relevant results based on the user's search parameters, helping users find mechanics that meet their needs.

Interactive features allow users to click on mechanic pins to view additional information such as contact details, service availability, and user ratings. Users can also use the map for route planning to the selected mechanic's location, with integration with navigation services for easy directions.

Real-time updates are provided on mechanic availability and service status. Mechanics can update their status, such as being available, busy, or offline, ensuring users have accurate and up-to-date information. . Users can also use the map for route planning to the selected mechanic's location, with integration with navigation services for easy directions. This comprehensive map feature improves the efficiency of locating and selecting mechanics, enhancing the overall user experience.

5.5 TESTING AND EVALUATION:

Testing and assessing the service is the last phase. To do this, user studies must be conducted, and the effectiveness must be assessed in terms of accuracy, responsiveness, and user happiness.

5.6 ADVANTAGES OF VAN-D: BREAKDOWN SYSTEM:

The Vehicle Breakdown Management System (VBMS) is designed to revolutionize the handling of vehicle breakdown incidents, particularly in projects involving fleets of vehicles. By leveraging technology, VBMS offers several advantages over traditional methods:

Efficient Incident Handling: VBMS streamlines the process of managing breakdown incidents, ensuring that mechanics are dispatched promptly to resolve issues, minimizing downtime.

Improved Response Times: With its ability to quickly locate nearby mechanics, VBMS significantly improves response times, getting vehicles back on the road faster.

Enhanced Communication: The system facilitates seamless communication between vehicle owners, mechanics, and administrators, ensuring that everyone is informed about the status of breakdown incidents.

Cost-Effectiveness: By reducing downtime and improving operational efficiency, VBMS helps save costs associated with vehicle breakdowns, making it a cost-effective solution.

Increased Customer Satisfaction: With quicker response times and efficient incident resolution, VBMS enhances customer satisfaction by minimizing the inconvenience caused by breakdowns.

Overall, VBMS is a comprehensive solution that not only improves the efficiency of managing breakdown incidents but also enhances customer satisfaction and reduces costs, making it a valuable tool for any project involving vehicle fleets.

5.7 CHALLENGES OF VAN-D: BREAKDOWN SYSTEM:

The On Road Vehicle Breakdown Assistance system and the Vehicle Breakdown Management System (VBMS) face several challenges that need to be addressed to ensure their effectiveness. One of the primary challenges is verifying the reliability and credentials of mechanics listed in the system to ensure that only trusted professionals are available to assist users. Additionally, implementing real-time location tracking for mechanics and vehicles requires integration with GPS systems and maintaining data accuracy.

Handling a high volume of service requests during peak times can strain the system, necessitating efficient dispatching and prioritization algorithms. Integrating the system with various platforms and devices used by mechanics, administrators, and customers is another challenge that requires compatibility testing and ongoing maintenance.

Ensuring the security and privacy of user data, including personal information and vehicle details, is crucial, especially with the need to comply

with data protection regulations.

As the number of vehicles and mechanics using the system grows, scalability becomes a key consideration to maintain performance. Finally, ensuring that all users, including mechanics and administrators, are trained to use the system effectively and adopt it into their workflow is essential for the system's success. Addressing these challenges requires careful planning, robust technology solutions, and effective communication between stakeholders.

5.8 TOOLS & TECHNOLOGIES:

Python 3.7.4

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.

It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

This tutorial gives enough understanding on Python programming language.

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable.

It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. We will list down some of the key advantages of learning Python:

Python is Interpreted - Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive - You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is a Beginner's Language - Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Data Science and Machine Learning with Python. The vast majority of the libraries used for data science or machine learning have Python interfaces, making the language the most popular high-level command interface for machine learning libraries and other numerical algorithms.

Web Services and RESTful APIs in Python

Python's native libraries and third-party web frameworks provide fast and convenient ways to create everything from simple REST APIs in a few lines of code to full-blown, data-driven sites. Python's latest versions have strong support for asynchronous operations, letting sites handle tens of thousands of requests per second with the right libraries.

Metaprogramming and Code Generation in Python

In Python, everything in the language is an object, including Python modules and libraries themselves. This lets Python work as a highly efficient code generator, making it possible to write applications that manipulate their own functions and have the kind of extensibility that would be difficult or impossible to pull off in other languages. Python can also be used to drive

code-generation systems, such as LLVM, to efficiently create code in other languages.

Python 2 vs Python 3

Python is available in two versions, which are different enough to trip up many new users. Python 2.x, the older “legacy” branch, will continue to be supported (that is, receive official updates) through 2020, and it might persist unofficially after that. Python 3.x, the current and future incarnation of the language, has many useful and important features not found in Python 2.x, such as new syntax features (e.g., the “walrus operator”), better concurrency controls, and a more efficient interpreter. Python 3 adoption was slowed for the longest time by the relative lack of third-party library support. Many Python libraries supported only Python 2, making it difficult to switch. But over the last couple of years, the number of libraries supporting only Python 2 has dwindled, all of the most popular libraries are now compatible with both Python 2 and Python 3.

Today, Python 3 is the best choice for new projects; there is no reason to pick Python 2 unless you have no choice. If you are stuck with Python 2, you have various strategies at your disposal.

Using an IDE

As good as dedicated program editors can be for your programming productivity, their utility pales into insignificance when compared to Integrated Developing Environments (IDEs), which offer many additional features such as in-editor debugging and program testing, as well as function descriptions and much more.

CHAPTER 6

SOURCE CODE

Manage.py:

```
import os
import sys
def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'vehicleservicemanagement.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)
if __name__ == '__main__':
    main()
```

Models.py:

```
from django.db import models
from django.contrib.auth.models import User
# Create your models here.
class Customer(models.Model):
    user=models.OneToOneField(User,on_delete=models.CASCADE)
    profile_pic=
```

```

models.ImageField(upload_to='profile_pic/CustomerProfilePic/',null=True,
blank=True)
    address = models.CharField(max_length=40)
    mobile = models.CharField(max_length=20,null=False)
    @property
    def get_name(self):
        return self.user.first_name+" "+self.user.last_name
    @property
    def get_instance(self):
        return self
    def __str__(self):
        return self.user.first_name
class Mechanic(models.Model):
    user=models.OneToOneField(User,on_delete=models.CASCADE)
    profile_pic=
models.ImageField(upload_to='profile_pic/MechanicProfilePic/',null=True,
blank=True)
    address = models.CharField(max_length=40)
    mobile = models.CharField(max_length=20,null=False)
    skill = models.CharField(max_length=500,null=True)
    salary=models.PositiveIntegerField(null=True)
    status=models.BooleanField(default=False)
    @property
    def get_name(self):
        return self.user.first_name+" "+self.user.last_name
    @property
    def get_id(self):
        return self.user.id

```

```

def __str__(self):
    return self.user.first_name
class Request(models.Model):
    cat=((('two wheeler with gear','two wheeler with gear'),('two wheeler
without gear','two wheeler without gear'),('three wheeler','three
wheeler'),('four wheeler','four wheeler'))
    category=models.CharField(max_length=50,choices=cat)
    vehicle_no=models.PositiveIntegerField(null=False)
    vehicle_name = models.CharField(max_length=40,null=False)
    vehicle_model = models.CharField(max_length=40,null=False)
    vehicle_brand = models.CharField(max_length=40,null=False)
    problem_description = models.CharField(max_length=500,null=False)
    date=models.DateField(auto_now=True)
    cost=models.PositiveIntegerField(null=True)
    customer=models.ForeignKey('Customer',
on_delete=models.CASCADE,null=True)
    mechanic=models.ForeignKey('Mechanic',on_delete=models.CASCADE,n
ull=True)
    stat=((('Pending','Pending'),('Approved','Approved'),('Repairing','Repairing'),
('Repairing Done','Repairing Done'),('Released','Released'))
    status=models.CharField(max_length=50,choices=stat,default='Pending',nu
ll=True)
    def __str__(self):
        return self.problem_description
class Attendance(models.Model):
    mechanic=models.ForeignKey('Mechanic',on_delete=models.CASCADE,n
ull=True)
    present_status = models.CharField(max_length=10)

```

Urls.py:

```
from django.contrib import admin
```

```
from django.urls import path
```

```
from vehicle import views
```

```
from django.contrib.auth.views import LoginView, LogoutView
```

```
urlpatterns = [
```

```
    path('admin/', admin.site.urls),
```

```
    path("", views.home_view, name=""),
```

```
    path('adminclick', views.adminclick_view),
```

```
    path('customerclick', views.customerclick_view),
```

```
    path('mechanicsclick', views.mechanicsclick_view),
```

```
    path('customersignup',
```

```
views.customer_signup_view, name='customersignup'),
```

```
    path('mechanicsignup',
```

```
views.mechanic_signup_view, name='mechanicsignup'),
```

```
    path('customerlogin',
```

```
LoginView.as_view(template_name='vehicle/customerlogin.html'), name='c
```

```
ustomerlogin'),
```

```
    path('mechaniclogin',
```

```
LoginView.as_view(template_name='vehicle/mechaniclogin.html'), name='
```

```
mechaniclogin'),
```

```
    path('adminlogin',
```

```
LoginView.as_view(template_name='vehicle/adminlogin.html'), name='ad
```

```
minlogin'),
```

```
    path('admin-dashboard', views.admin_dashboard_view, name='admin-  
dashboard'),
```

```
    path('admin-customer', views.admin_customer_view, name='admin-
```

```

customer'),
    path('admin-view-
customer',views.admin_view_customer_view,name='admin-view-
customer'),
    path('delete-customer/<int:pk>',
views.delete_customer_view,name='delete-customer'),
    path('update-customer/<int:pk>',
views.update_customer_view,name='update-customer'),
    path('admin-add-customer',
views.admin_add_customer_view,name='admin-add-customer'),
    path('admin-view-customer-enquiry',
views.admin_view_customer_enquiry_view,name='admin-view-customer-
enquiry'),
    path('admin-view-customer-invoice',
views.admin_view_customer_invoice_view,name='admin-view-customer-
invoice'),
    path('admin-request', views.admin_request_view,name='admin-request'),
    path('admin-view-
request',views.admin_view_request_view,name='admin-view-request'),
    path('change-status/<int:pk>', views.change_status_view,name='change-
status'),
    path('admin-delete-request/<int:pk>',
views.admin_delete_request_view,name='admin-delete-request'),
    path('admin-add-request',views.admin_add_request_view,name='admin-
add-request'),
    path('admin-approve-
request',views.admin_approve_request_view,name='admin-approve-
request'),

```

```

    path('approve-request/<int:pk>',
views.approve_request_view,name='approve-request'),
    path('admin-view-service-
cost',views.admin_view_service_cost_view,name='admin-view-service-
cost'),
    path('update-cost/<int:pk>', views.update_cost_view,name='update-
cost'),
    path('admin-mechanic', views.admin_mechanic_view,name='admin-
mechanic'),
    path('admin-view-
mechanic',views.admin_view_mechanic_view,name='admin-view-
mechanic'),
    path('delete-mechanic/<int:pk>',
views.delete_mechanic_view,name='delete-mechanic'),
    path('update-mechanic/<int:pk>',
views.update_mechanic_view,name='update-mechanic'),
    path('admin-add-
mechanic',views.admin_add_mechanic_view,name='admin-add-mechanic'),
    path('admin-approve-
mechanic',views.admin_approve_mechanic_view,name='admin-approve-
mechanic'),
    path('approve-mechanic/<int:pk>',
views.approve_mechanic_view,name='approve-mechanic'),
    path('delete-mechanic/<int:pk>',
views.delete_mechanic_view,name='delete-mechanic'),
    path('admin-view-mechanic-
salary',views.admin_view_mechanic_salary_view,name='admin-view-
mechanic-salary'),

```



```

    path('update-salary/<int:pk>', views.update_salary_view,name='update-
salary'),
    path('admin-mechanic-attendance',
views.admin_mechanic_attendance_view,name='admin-mechanic-
attendance'),
    path('admin-take-attendance',
views.admin_take_attendance_view,name='admin-take-attendance'),
    path('admin-view-attendance',
views.admin_view_attendance_view,name='admin-view-attendance'),
    path('admin-feedback', views.admin_feedback_view,name='admin-
feedback'),
    path('admin-report', views.admin_report_view,name='admin-report'),
    path('mechanic-dashboard',
views.mechanic_dashboard_view,name='mechanic-dashboard'),
    path('mechanic-work-assigned',
views.mechanic_work_assigned_view,name='mechanic-work-assigned'),
    path('mechanic-update-status/<int:pk>',
views.mechanic_update_status_view,name='mechanic-update-status'),
    path('mechanic-feedback',
views.mechanic_feedback_view,name='mechanic-feedback'),
    path('mechanic-salary', views.mechanic_salary_view,name='mechanic-
salary'),
    path('mechanic-profile', views.mechanic_profile_view,name='mechanic-
profile'),
    path('edit-mechanic-profile',
views.edit_mechanic_profile_view,name='edit-mechanic-profile'),
    path('mechanic-attendance',
views.mechanic_attendance_view,name='mechanic-attendance').

```

```

    path('customer-dashboard',
views.customer_dashboard_view,name='customer-dashboard'),
    path('customer-request', views.customer_request_view,name='customer-
request'),
    path('customer-add-
request',views.customer_add_request_view,name='customer-add-request'),
    path('customer-profile', views.customer_profile_view,name='customer-
profile'),
    path('edit-customer-profile',
views.edit_customer_profile_view,name='edit-customer-profile'),
    path('customer-feedback',
views.customer_feedback_view,name='customer-feedback'),
    path('customer-invoice', views.customer_invoice_view,name='customer-
invoice'),
    path('customer-view-
request',views.customer_view_request_view,name='customer-view-
request'),
    path('customer-delete-request/<int:pk>',
views.customer_delete_request_view,name='customer-delete-request'),
    path('customer-view-approved-
request',views.customer_view_approved_request_view,name='customer-
view-approved-request'),
    path('customer-view-approved-request-

```

Views.py

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-
scale=1.0">

    <title>Realtime Location Tracker with Excel Data</title>

    <!-- leaflet css -->

    <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />

    <style>
        body {
            margin: 0;
            padding: 0;
        }
        #map {
            width: 100%;
            height: 80vh; /* Adjust the height as needed */
        }
    </style>
</head>

<body>

    <div id="map"></div>

</body>

<!-- leaflet js -->

<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
```

```

<script>
    // Map initialization
    var map = L.map('map').setView([14.086, 100.608], 6);
    var osm =
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
    });
    osm.addTo(map);
    // Excel data converted to JSON
    var jsonData = [
        { "lat": 14.086, "lng": 100.608, "name": "Marker 1" },
        { "lat": 13.756, "lng": 100.501, "name": "Marker 2" }
        // Add more data as needed
    ];
    // Create markers from JSON data
    jsonData.forEach(function(data) {
        var marker = L.marker([data.lat, data.lng]).addTo(map);
        marker.bindPopup(data.name);
    });
    // Geolocation code from the second code snippet
    var marker, circle;
    var reboundTimer;
    function getPosition(position) {
        var lat = position.coords.latitude;
        var long = position.coords.longitude;
        var accuracy = position.coords.accuracy;

```

```

    if (marker) {
        map.removeLayer(marker);
    }
    if (circle) {
        map.removeLayer(circle);
    }
    marker = L.marker([lat, long]);
    circle = L.circle([lat, long], { radius: accuracy });
    var featureGroup = L.featureGroup([marker, circle]).addTo(map);
    clearTimeout(reboundTimer); // Clear any existing rebound timer
    reboundTimer = setTimeout(function() {
        map.fitBounds(featureGroup.getBounds());
    }, 2000090000); // 2 second delay
    console.log("Your coordinate is: Lat: " + lat + " Long: " + long + "
Accuracy: " + accuracy);
}
if (!navigator.geolocation) {
    console.log("Your browser doesn't support geolocation feature!");
} else {
    setInterval(() => {
        navigator.geolocation.getCurrentPosition(getPosition);
    }, 5000);
}
</script>
</html>

```

Settings.py:

```
from django.core.mail import send_mail
import os
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
TEMPLATE_DIR = os.path.join(BASE_DIR, 'templates')
STATIC_DIR = os.path.join(BASE_DIR, 'static')
UTILS_DIR = os.path.join(BASE_DIR, 'utils')
MEDIA_ROOT = os.path.join(BASE_DIR, 'static')
#hi
LOGIN_URL = '/accounts/login/'
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'ftxnh_7475z^joy_*l9t*qnqow!@)y#(541^w1=(8--
=3g#4*d'
# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
```

```

'django.contrib.staticfiles',
'vehicle',
'widget_tweaks',
]
MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'vehicleservicemanagement.urls'
TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [TEMPLATE_DIR,],
'APP_DIRS': True,
'OPTIONS': {
'context_processors': [
'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
},
],

```

```

]
WSGI_APPLICATION = 'vehicleservicemanagement.wsgi.application'

# Database

# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

# Password validation

# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator'
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':

```



```

'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
    {
    }
]

# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/
LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'

STATICFILES_DIRS=[
    STATIC_DIR,
]

LOGIN_REDIRECT_URL='/afterlogin'

```

```
#for contact us give your gmail id and password
EMAIL_BACKEND='django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST='smtp.gmail.com'
EMAIL_USE_TLS=True
EMAIL_PORT=587
EMAIL_HOST_USER='727721epci009@skcet.ac.in' # this email will be
used to send emails
EMAIL_HOST_PASSWORD='skcet123' # host email password required
# now sign in with your host gmail account in your browser
# open following link and turn it ON
# https://myaccount.google.com/lesssecureapps
# otherwise you will get SMTPAuthenticationError at /contactus
# this process is required because google blocks apps authentication by
default
EMAIL_RECEIVING_USER=['727721epci054@skcet.ac.in'] # email on
which you will receive messages sent from website
```

Folium.html:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Realtime Location Tracker with Excel Data</title>
```

```

<!-- leaflet css -->
<link rel="stylesheet"
href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />
<style>
  body {
    margin: 0;
    padding: 0;
  }

  #map {
    width: 100%;
    height: 80vh; /* Adjust the height as needed */
  }
</style>
</head>

<body>
  <div id="map"></div>
</body>

<!-- leaflet js -->
<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"></script>
<script>
  // Map initialization
  var map = L.map('map').setView([14.086, 100.608], 6);

  var osm =
  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

```

```

        attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
    });
    osm.addTo(map);

// Excel data converted to JSON
var jsonData = [
    { "lat": 14.086, "lng": 100.608, "name": "Marker 1" },
    { "lat": 13.756, "lng": 100.501, "name": "Marker 2" }
    // Add more data as needed
];

// Create markers from JSON data
jsonData.forEach(function(data) {
    var marker = L.marker([data.lat, data.lng]).addTo(map);
    marker.bindPopup(data.name);
});

// Geolocation code from the second code snippet
var marker, circle;
var reboundTimer;

function getPosition(position) {
    var lat = position.coords.latitude;
    var long = position.coords.longitude;
    var accuracy = position.coords.accuracy;

```

```

    if (marker) {
        map.removeLayer(marker);
    }

    if (circle) {
        map.removeLayer(circle);
    }
    marker = L.marker([lat, long]);
    circle = L.circle([lat, long], { radius: accuracy });
    var featureGroup = L.featureGroup([marker, circle]).addTo(map)
    clearTimeout(reboundTimer); // Clear any existing rebound timer
    reboundTimer = setTimeout(function() {
        map.fitBounds(featureGroup.getBounds());
    }, 2000090000); // 2 second delay
    console.log("Your coordinate is: Lat: " + lat + " Long: " + long + "
Accuracy: " + accuracy);
}
if (!navigator.geolocation) {
    console.log("Your browser doesn't support geolocation feature!");
} else {
    setInterval(() => {
        navigator.geolocation.getCurrentPosition(getPosition);
    }, 5000);
}

</script>

</html>

```

CHAPTER 7

SCREENSHOT

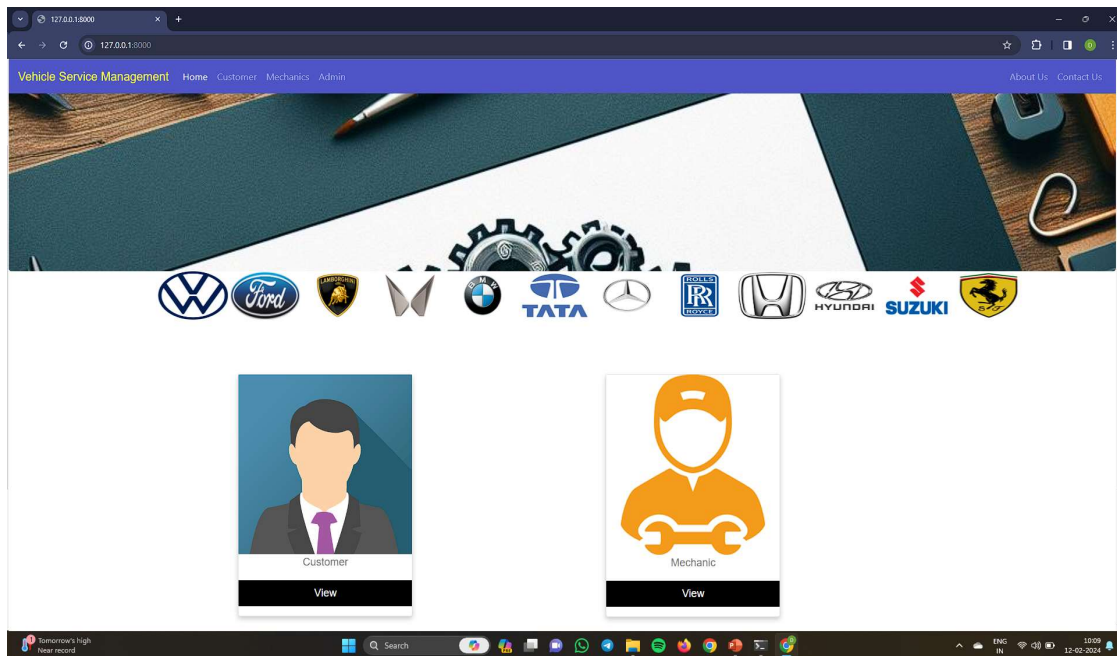


Fig 7.1 Home page

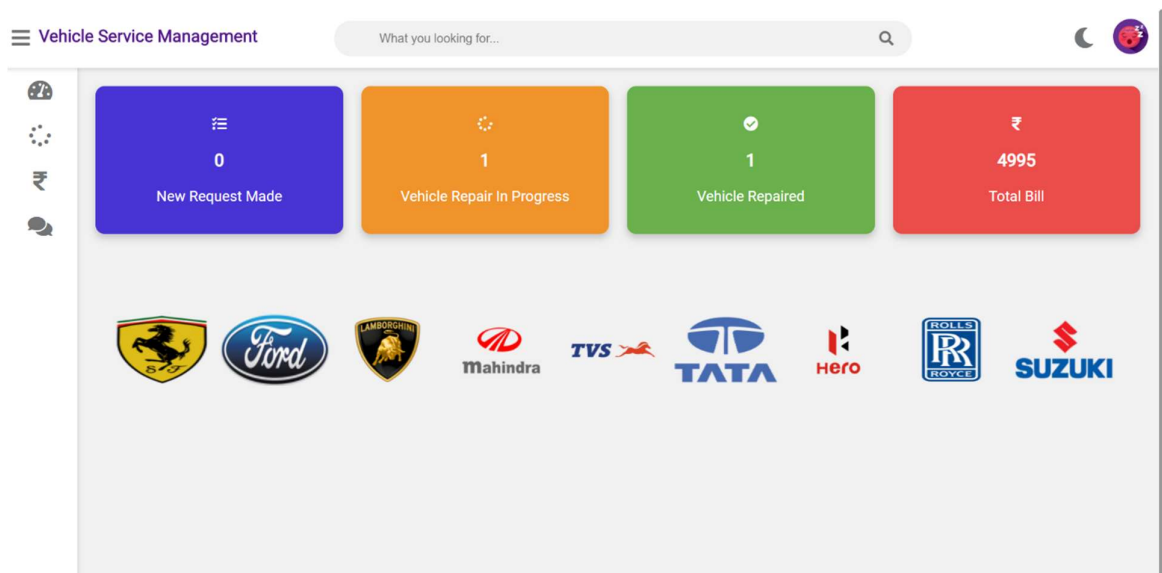


Fig 7.2 Customer dashboard

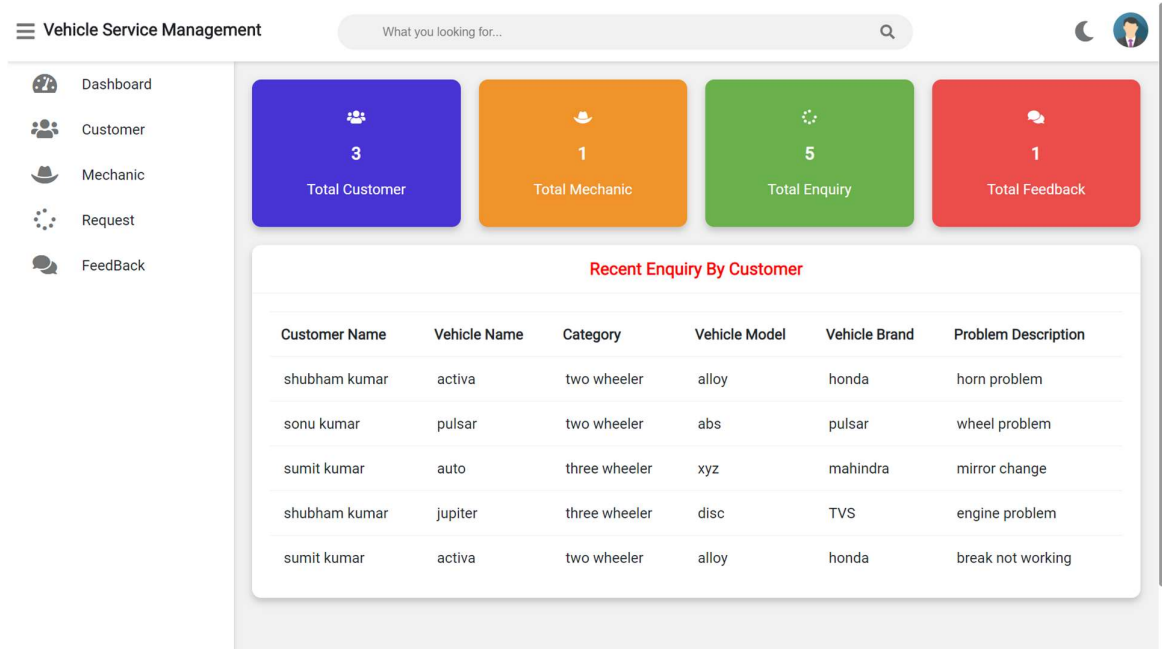


Fig 7.3 Admin dashboard

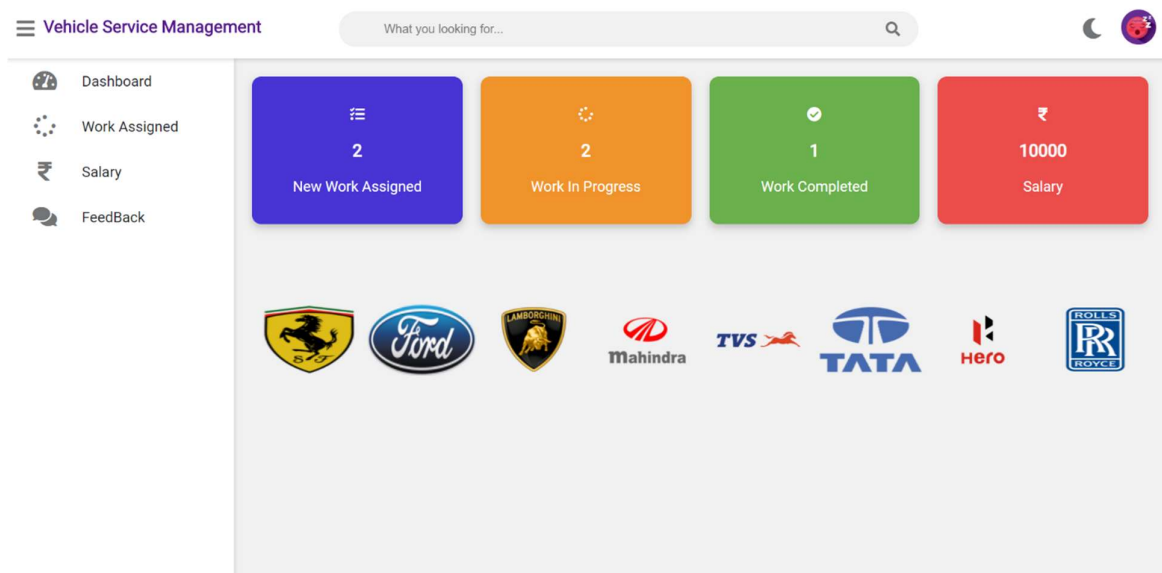


Fig 7.4 Mechnaic dashboard

Approved Request With Cost

Vehicle Name	Vehicle Number	Problem Description	Enquiry Date	Total Cost	
Splendor	6362	starting prblm	Feb. 2, 2024	500	Pay
dio	2792	headlight prblm	Feb. 10, 2024	800	Pay
honda	2121	starting prblm	March 21, 2024	300	Pay

Fig 7.5 Invoice Page

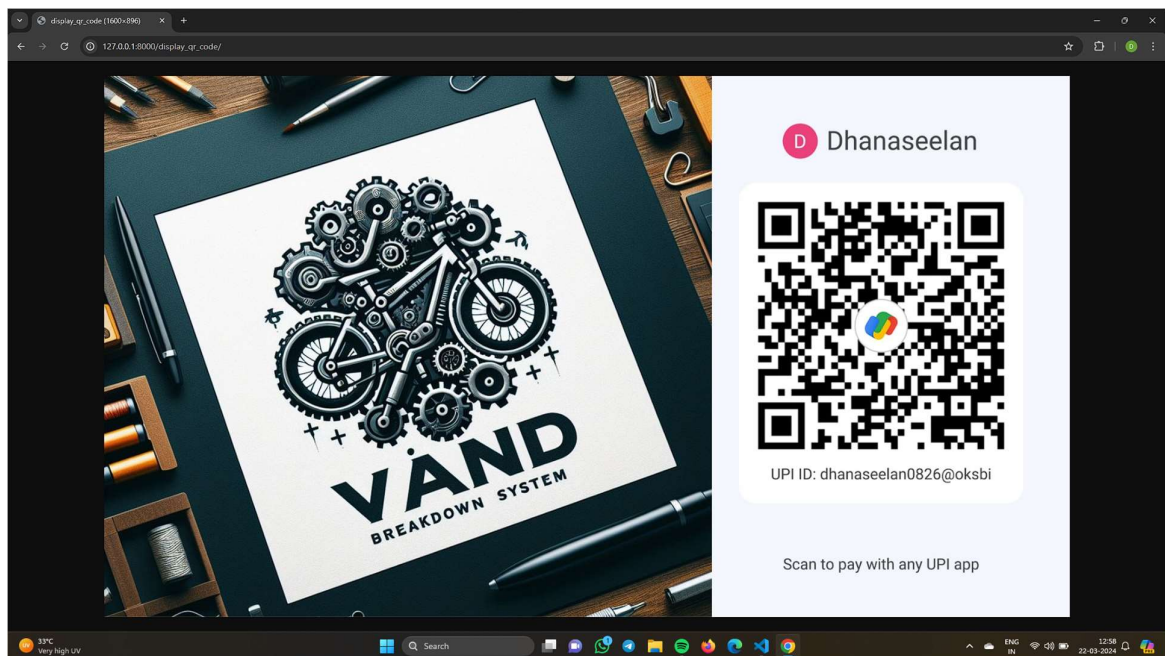


Fig 7.6 Payment Page

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 CONCLUSION:

In conclusion, the On Road Vehicle Breakdown Assistance system and the Vehicle Breakdown Management System (VBMS) are comprehensive solutions designed to streamline and enhance the efficiency of handling vehicle breakdown incidents.

Overall, the On Road Vehicle Breakdown Assistance system and the Vehicle Breakdown Management System represent significant advancements in the field of vehicle breakdown assistance, offering innovative solutions to improve the efficiency and reliability of roadside assistance services.

8.2 FUTURE SCOPE:

The future of On Road Vehicle Breakdown Assistance systems and Vehicle Breakdown Management Systems (VBMS) holds exciting possibilities for advancements and improvements. One of the key areas of development is the integration of AI and machine learning. By implementing advanced algorithms, these systems can enhance predictive maintenance capabilities and optimizing resource allocation for faster response times.

Enhancing the user experience is another important focus for future systems. This could involve improving user interfaces, providing more interactive maps, and integrating with virtual assistants for a more intuitive and user-friendly experience. These systems leverage a variety of tools and techniques, including Geographic Information System (GIS) technology, Global Positioning System (GPS) technology, database management systems, data analytics tools, and machine learning algorithms are used.

REFERENCES

- [1]Ms.M.Nivetha, Mrs.S.Sujitha ,Ms. V.Abinaya.Vehicle breakdown Assistance . D. Angeline, A. Arul et al.(Eds): ICICI-2021 pp. 193-202, 2021. IJCI – 2021 DOI:10.5121/ijci.2021.100221
- [2]Artigala, V., & Nadeeshani, H. D. Recent Advances in Connected Vehicle Technologies for Roadside Assistance Systems. IEEE Intelligent Transportation Systems Magazine (2019)
a. DOI:10.5121/ijci.2019.100221
- [3]“Abraham Sudharson Ponraj, Shivang Shah, Parimal Abhishek, Deep Shrivastava”, “Vehicle Service Management and Live Monitoring With Predictive Maintenance System”, 2019 International Conference on Vision Towards Emerging
- [4]Iswarya, K., Devaki, D., & Ranjith, E. C o m p a r a t i v e A n a l y s i s o f M o b i l e Applications for Vehicle Breakdown Assistance in Urban Areas. I E E E A c c e s s b y M . A . Naeem (2018) DOI:10.57238/ieee.2018.100323
- [5]“Hanamant B. Sale, Dharmendra Bari, TanayDalvi, Yash Pandey”, “Online Management System for Automobile Services”, International Journal of Engineering Science and Computing (IJESC), Volume 8 Issue No.02, March-2018.
- [6]Khanapuri, A. V., Shastri, A., Dsouza, G., & Dsouza, S. A Critical Review of Onboard Diagnostics (O B D) S y s t e m s f o r P r e d i c t i v e Maintenance and Breakdown Prevention IEEE Transactions on Intelligent Transportation Systems (2017) DOI:10.5121/ijci.2017.100221

[7]“Prof. Shilpa Chavan Saket Adhav, Rushikesh Gujar, Mayur Jadhav, Tushar Limbore”, “Automobile Service Center Management System”, International Journal of Scientific and Research Publications, Volume 4, Issue 3, March 2014.

[8]Anas Siddiqui, Usman Saleem, Abdul Ur Rehman and Shiraz Latif Sohaib, "GPS and GSM Based Advanced Vehicle Monitoring and Information System", First International Conference on Modern Communication Computing Technologies (MCCT14).

[9]JinSheng, K., Bahurudin, A. S., & Karkonasasi, K. Fault Detection and Location in Electric Vehicles Using Data Analytics and Machine Learning Imaging for Crime Detection and Prevention (ICDP 2013 DOI:10.5121/ijci.2013.100221

[10]Jr-Jen Huang, Yi-Yu Chu and Yen-Jen Chen, "The System Design and Implementation of Vehicle Management", Journal of Advances in Computer Networks, vol. 1, no. 1, March 2013.

[11]“Doctoral Dissertation by Jonas Kuschel”, “Vehicle Services”, Studies in Applied Information Technology, September 2009 ISSN 1652-490X;7, ISBN 978-91-628-7870-2

[12]Phongsak Keeratiwintakorn, "Real-Time Tracking Management System Using GPS GPRS and Google Earth", June 2008.

[13]J. Parthasarathy, "POSITIONING AND NAVIGATION SYSTEM USING GPS" in International Archives of the Photogrammetry Remote Sensing and Spatial Information Science, Tokyo Japan, vol. XXXVI, 2006.