

RIDE RIGHT

Submitted in partial fulfilment of the required for the award of
Bachelor of Computer Science

By

POOJA C
(REGISTER NO 41290042)

SUMITHRA S
(REGISTER NO 41290059)



DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF SCIENCE AND HUMANITIES

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

Accredited with Grade “A++” by NAAC

JEPPIAR NAGAR, RAJIV GANDHI SALAI, CHENNAI 600 119

APRIL 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

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

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **SUMITHRA S (41290059)** and **POOJA C (41290042)** who carried out the project entitled "**Ride Right**" under our supervision from February 2024 TO April 2024.

Internal Guide

Ms. ANCY A, M.Sc., Assistant Professor

Dean And Head of The Department

Dr. REKHA CHAKRAVARTHI M.E., Ph.D.

Submitted for Viva voce examination held on

27/04/2024

Internal Examiner

External Examiner

DECLARATION

We **SUMITHRA S(41290059)** and **POOJA C (41290042)** hereby declare that the Project Report entitled “**Ride Right**” done by under the guidance of **Ms. ANCY A, M.Sc., Assistant Professor** Department of Computer Science at **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY**, Jeppiar Nagar, Rajiv Gandhi Salai, Chennai-600 119 is submitted in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

DATE: 27/04/2024

PLACE: CHENNAI

SUMITHRA S

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

We are pleased to acknowledge our sincere thanks to Board of Management **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. We grateful to them.

We convey our thanks to Dr. REKHA CHAKRAVARTHI M.E., Ph.D. Dean of **Arts and Science** for providing us necessary support and details at the right time during the progressive reviews.

I would like to express our sincere and deep sense of gratitude to our project guide **Ms. ANCY A, M.Sc., Assistant Professor** Department of Computer Science for their valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

We wish to express my thanks to all Teaching and Non-teaching staff member of the Department of Computer Science who were helpful in many ways for the completion of the project

SUMITHRA S

ABSTRACT

When a person's car experiences mechanical problems, the Ride Right (RR) will be an appropriate response for them to seek assistance in faraway locations. The chosen public will be the RR's clients, and they will connect with the particular expert through a reliable application process. The Ride Right (RR) addresses the challenge of stranded drivers experiencing car trouble on the road. Ride Right assistance system that connects users with qualified mechanics who can travel to their location for repairs. The system will function through a mobile application. Users experiencing a breakdown can utilize the app to report their location, vehicle details, and the nature of the problem. The application will then leverage geolocation to identify nearby registered mechanics with the expertise to address the issue. This project aims to provide a convenient, efficient, and reliable solution for roadside breakdowns. It offers peace of mind to drivers by ensuring prompt and qualified assistance, ultimately enhancing safety and minimizing disruption during unexpected car troubles.

TABLE OF CONTENT

CH.NO	NAME OF THE TOPIC	PAGE NO.
1	INTRODUCTION	1
1.1	Overview	1
1.2	Problem Statement	2
1.3	Objective	2
1.4	Literature Review	3
2	SYSTEM ANALYSIS	5
2.1	System Analysis Description	5
2.2	Existing System	5
2.3	Proposed System	5
3	SYSTEM REQUIREMENTS	7
3.1	Software Requirements	7
3.2	Hardware Requirements	8
3.3	Introduction to JavaScript Programming	9
3.4	React native / Node JS	10
3.5	Mongo DB	11
4	SYSTEM DESIGN	13
4.1	System Architecture	15
4.2	DFD Diagram	18
4.3	ER Diagram	20

5	SYSTEM IMPLEMENTATION	23
5.1	Model Description	23
5.2	Sample code	28

CH.NO	NAME OF THE TOPIC	PAGENO.
6	SYSTEM TESTING	31
6.1	Methodology	31
7	CONCLUSION	33
8	FUTURE ENHANCEMENT AND FEATURES	34
9	APPENDIX	36
9.1	Coding	36
9.2	Screenshots	42
10	REFERENCE	46

LIST OF FIGURES

FIG.NO	LIST OF THE FIGURES	PAGE NO
9.1.1	User Login	46
9.1.2	Mechanic Login	47
9.1.3	User Request	48
9.1.4	Alert Message	49
9.1.5	Mechanic Home	49
9.1.6	Request Status	49

CHAPTER 1

INTRODUCTION

A lot of individuals face difficulties getting help when their vehicle breaks down on the road. Many of them do not have any automobile repair provider within their reach and couldn't get assist because the auto repair provider middle is perhaps far away from their place. This venture to help folks who are in want whilst their automobile breaks down alongside the roads to make this viable, we expand Android Application for consumer help which affords help to the passengers at some stage in their street trips. We recognize how many a way Android has imparted to the arena of technology. The very truth that Android is open source has resulted with inside the OS being imparted in smartphones with essential hardware. This in turn has resulted with inside the world of net being accessed via way of means of humans even with inside the remotest of locations and henceforth we are going to use this utility which can be utilized by humans below misery of a breakdown and mix the various feasible help that will be furnished at some point of the trip.

1.1 OVERVIEW

The Ride Right project aims to revolutionize roadside assistance. Imagine a system that uses GPS and diagnostics to automatically detect a car breakdown. The system would then identify the issue, locate nearby service providers, and connect the driver for help. This project could leverage mobile apps for user interaction, allowing drivers to share location, vehicle information, and receive real-time updates. Potential benefits include reduced stress during breakdowns, faster response times, and improved safety on the road. However, challenges include developing accurate diagnostics, ensuring network connectivity, and building partnerships with service providers. Overall, this project has the potential to significantly improve the roadside assistance experience for drivers.

1.2 PROBLEM STATEMENT

Vehicle breakdowns can be stressful and inconvenient, especially when they occur in unfamiliar locations or at night. Traditional methods of finding roadside assistance often involve unreliable phone calls, limited towing options, or a lack of transparency in pricing and service quality. This project aims to address these challenges by developing a comprehensive on-road vehicle

breakdown. However, several key problems need to be tackled. Firstly, users often struggle to quickly locate qualified mechanics in their immediate vicinity, particularly in remote areas. Secondly, a lack of clear communication regarding the nature of the breakdown and the expertise required can lead to delays and mismatched services. Thirdly, concerns arise about user safety when stranded on the roadside, with limited information about the estimated arrival time of assistance or the qualifications of the dispatched mechanic. Furthermore, the current system can be opaque in terms of pricing, leading to anxieties about hidden fees or unfair charges. Additionally, existing solutions may not cater to a diverse range of vehicle types or offer support for breakdowns requiring specialized towing or recovery services. Finally, the user experience can be fragmented, requiring phone calls, separate apps for payment, and potentially managing interactions with multiple parties involved in the resolution process. This project seeks to create a seamless, user-friendly platform that addresses these pain points by leveraging features such as location-based services to connect users with verified mechanics, facilitating clear communication about the breakdown and service needs, and providing transparent pricing estimates upfront. The system should prioritize user safety by ensuring timely assistance and verified credentials of service providers. Additionally, the platform should encompass a wider range of vehicle types and offer specialized breakdown solutions. Finally, the project aims to streamline the user experience by integrating all functionalities, including requesting assistance, communication, payment, and service tracking, within a single, user-friendly interface. By tackling these problems, this on-road vehicle breakdown assistant system aspires to transform a stressful and uncertain situation into a swift, efficient, and reliable experience for drivers.

1.3 OBJECTIVES

The objective of the Ride Right project is to empower drivers stranded with car trouble. This user-friendly system, envisioned as a mobile app or similar platform, aims to bridge the gap between stranded motorists and reliable mechanics. By leveraging location services, the project will connect users to qualified mechanics in their vicinity, allowing them to request assistance, receive quotes, and potentially schedule repairs – all from the convenience of their smartphones. This not only reduces stress and wait times during a stressful situation, but also fosters a sense of security, especially for those unfamiliar with the area or lacking a trusted repair shop.

1.4 LITRATURE SURVEY

S.NO	Author	Title	Year
1.	WangW., Chen H., and Bell M.	Vehicle Breakdown Duration Modelling	2005
2.	Haridas K, Ahmad S. Baharudin.	Automotive Servicing and Breakdown Assistance System (ASBAS).	2015
3.	Kamal & karkonasasi	Vehicle Breakdown Servicing Necessity (VBSN)	2016
4.	Kapadi V., Guruju S., & Bojja B.	Emergency Breakdown Services using Android Application	2017
5.	Kumaar.A, Balakrishna,Subha. S, Harin.	On Road Vehicle Service finder	2019
6.	Miss K Iswarya, Miss. D. Devika, and MR. E. Ranjith.	Road Assistance System using GPS .	2010
7.	Reichardt.	vehical talk 2000	2018

DESCRIPTION

The recent technological advancement in the field of mobility has impressed the current society tremendously. (Haridas, Baharudin, & karkonasasi ,2016) Based on AAM annual report, statistic shows that around 70% of service is resolved immediately which is classified as minor failure breakdown issue. Minor failure breakdown has segregated into several categories like engine failure start up, engine failure heat,lockout, and others. This statistic is only from single organization apart from the other services available nationwide. This clearly shows that the number of vehicles encountering such emergency situation requires the service to be available. (Haridas, Baharudin, & karkonasasi ,2016) Firstly, the existing manual process is not able to address the transaction request effectively. The manual process is holding back staff as they are unable to provide quick response due to lack of streamlined system with sufficient information available. Secondly, quick responses features are insufficient in the existing products. This limits a quick response for the consumer when there is an emergency request. In the existing products, the emergency request is being handled by the call center hotline as an interim person before reaching the vendors. This causes the consumers to experience slow responses due to long waiting hours at the call center. (Kumar.D, Balakrishna, Subha, Harin. K, 2019) Akhila V Khanapuri et al (2015) proposed that there has been an exponential increase in the number of cars on road, number of road accidents and vehicle breakdown cases recorded. Finding effective ways to achieve maximum fuel efficiency without hampering the internal structure of these vehicles along with providing a response system to combat mishaps is a challenging task. Main objective is to design a system that can be easily installed and to provide platform for fast access.

CHAPTER 2

SYSTEM ANALYSIS

The Ride Right Vehicle Breakdown Assistant system analysis dissects the project's components and functionalities. Here, we'll examine user needs, system requirements, and potential challenges. User needs likely involve a user-friendly interface for requesting assistance, real-time mechanic location, and secure payment options. System requirements encompass a mobile app with GPS integration, a database of verified mechanics, and a communication platform for user-mechanic interaction. Challenges might include ensuring comprehensive mechanic coverage across diverse locations, maintaining data security for user and mechanic information, and developing a robust platform that functions flawlessly even in areas with weak network connectivity.

2.1 SYSTEM ANALYSIS DESCRIPTION

The Driver breakdown requests with location and details flow to the system, which matches them with available mechanics based on location and potentially user ratings/expertise. Secure communication facilitates interaction between drivers and mechanics. Administrators manage the system's health, analysis data for service optimization, and ensure quality control. Drivers experience reduced stress through quick and reliable assistance, potentially including estimated repair costs and secure payment options. Mechanics benefit from optimized workload with local requests and appointment management tools. The system offers a potential revenue stream for monetization.

2.2 EXISTING SYSTEM

With the present framework, they can use just getting the current area to find the location. That suggests that additional software is required to discern the nearby area, which has various drawbacks. That is why we need to specify the location so that it can search the neighbourhood for the items we've asked for

2.3 PROPOSED SYSTEM

The Our proposed-crisis breakdowns administration which gives the easy to understand climate. is most straightforward way to character the area as well as neighboring required-areas our application gives close by area required areas our application gives close by area data, for example,

hospitals. We can also add our family details-mobile numbers, so that when an accident occurs a click on the send alert message or by shaking the mobile sends the current location of the vehicle to the family members contact number. The Provokes involved when the vehicle stops in unforeseen conditions are quite a large number. In such an unfriendly circumstance, client may not precisely know the area to view as the closest accessible specialist. The objective is to foster an Android application that will assist the client with benefiting help by introducing the application and gain admittance to the close by specialist and accordingly reach them utilizing the web office and the application. Overall, when the client is struck in an especially ominous situation, the underlying step embraced is to enquire with people nearby and find the region of the nearest subject matter expert and get the support cycle working. The proposed framework created is an Android put together application which can run with respect to any of theviable Android gadgets, be it a Tablet or a Cell phone. The application will empower the vehicle proprietor to look and speak with any assistance place's technician nearby. The client gains admittance to the technician's area and the repairman consequently gains admittance to the client's area, through the application utilizing the google map office subsequently saving a great deal of time in such a muddled circumstance Hereafter, the proposed framework remains closely connected with the new age innovation and describes - ease of use, education and efficient

CHAPTER 3

SYSTEM CONFIGURATION

The Ride Right Breakdown Assistant will be a mobile application with a three-tier architecture. The user interacts with a user-friendly mobile app (front-end). This app connects to a secure cloud-based backend that stores user data, mechanic profiles, and breakdown requests. The backend leverages GPS data and user input to match drivers with nearby mechanics. Finally, an admin panel (web interface) allows administrators to manage user registrations, mechanic credentials, and system health. This cloud-based system ensures scalability and accessibility for all users.

3.1 SOFTWARE REQUIREMENTS

The Front-End (Mobile App):

React Native: A JavaScript framework for building cross-platform mobile apps (iOS and Android). It allows you to write code once and deploy it to both platforms, saving development time.

JavaScript (ES6+): The primary programming language for React Native development.

React Native Navigation A popular library for creating navigation structures within your React Native app. This helps users move between different screens in the app.

Maps Integration: Google Maps Platform Offers map functionalities like location display, directions, and nearby place search. Requires setting up a Google Cloud Platform project and obtaining an API key. Apple Maps (iOS only) Provides similar functionalities as Google Maps but is specific to iOS devices.

In-App Chat Library: Firebase Offers real-time chat functionality with features like message history and push notifications. Requires setting up a Firebase project and integrating with the app. Socket.IO Another option for real-time communication, allowing for more customization but requiring more backend development effort. **Payment Gateway Integration** **Stripe** A popular payment processing platform that allows users to securely pay for services within the app. Requires a Stripe account and API integration. **PayPal** Another widely used payment gateway offering similar functionalities to Stripe.

Backend (Node.js): Node.js JavaScript runtime environment that allows you to run JavaScript code outside of a web browser. This enables server-side development for your application. Express.js A popular web framework for building APIs in Node.js. It simplifies creating routes, handling requests and responses, and managing middleware. Mongoose An Object Document Mapper (ODM) library for MongoDB that provides a more intuitive way to interact with the database using JavaScript objects. Database Driver A driver library is needed to connect Node.js to MongoDB. The official MongoDB Node.js Driver is recommended.

Database: A NoSQL document database that stores data in flexible JSON-like documents. This allows for schema-less data structures, making it well-suited for storing user information, mechanic profiles, and breakdown requests.

3.2 HARDWARE REQUIREMENTS

The Processor: An Intel 11th gen i5 processor is a powerful choice, but for development purposes, a more recent Intel Core i3 or AMD Ryzen 3 processor would be perfectly adequate. These processors offer enough processing power to handle the development tools and run the app simulator effectively.

Hard Disk: While 1TB of storage is great, a typical project wouldn't require that much space. A 256GB Solid State Drive (SSD) would be ideal for fast development cycles and program loading. SSDs offer significant speed advantages over traditional HDDs.

RAM: 1TB of RAM is far more than necessary. 8GB of RAM is a good starting point for development, and 16GB would be very comfortable for multitasking and running resource-intensive tools.

Monitor: A 15-inch monitor, either CRT or LCD, will work, but a larger display (at least 24 inches) would provide more screen real estate for development tools and code editing. Resolution is also important - Full HD (1920x1080) or higher is recommended for better clarity.

Mouse: Any standard USB mouse will suffice. For extended coding sessions, a comfortable ergonomic mouse might be a good investment.

3.3 INTRODUCTION TO JAVASCRIPT PROGRAMMING

JavaScript (JS) is a high-level, interpreted programming language that underpins interactivity on the web. It enables web pages to be dynamic and responsive to user actions, creating a more engaging user experience.

History:

Invented by Brendan Eich in 1995 at Netscape, initially named "LiveScript." Designed to add interactivity to static web pages, making them more dynamic and engaging. Quickly became the standard scripting language for web browsers.

Key Features:

Interpreted Language No separate compilation step needed, code is executed line by line. Lightweight Easy to learn and use, runs efficiently in web browsers. Dynamically Typed Data types are not declared upfront, offering flexibility. Prototypal Inheritance Objects inherit properties and methods from other objects. Event-Driven Responds to user actions and browser events, making web pages interactive. Single-Threaded Executes code one line at a time, but asynchronous techniques can handle long-running tasks.

Versatility and Libraries:

Extends beyond web development used for server-side scripting (Node.js), desktop apps (Electron), and mobile development (React Native). Rich ecosystem of libraries and frameworks jQuery (DOM manipulation), React (UI components), Angular (full-fledged framework), and many more.

Cross-Platform Compatibility:

Runs on all major web browsers, ensuring consistent user experience across platforms. Community and Ecosystem Large and active developer community offering support, tutorials, and continuous language evolution through standards bodies (ECMAScript). Extensive online resources, libraries, and frameworks available for various development needs.

Why JavaScript Matters:

Essential for Web Development Dominant language for building interactive web applications and single-page applications (SPAs). Versatile and Adaptable Used across different programming paradigms and platforms. Relatively Easy to Learn Beginner-friendly syntax and vast learning resources. High Demand JavaScript developers are highly sought-after in the tech industry.

Productivity:

JavaScript frameworks like React and Angular streamline development by providing pre-built components and functionalities. Abundant libraries handle common tasks like data manipulation, animations, and user interface interactions, saving development time.

3.4 REACT NATIVE/ NODE JS

React Native:

Cross-Platform Mobile App Development Create native-looking mobile apps for iOS and Android using a single JavaScript codebase. This saves development time and resources compared to building separate native apps for each platform. JavaScript Expertise Leverage existing JavaScript skills to build mobile apps, reducing the learning curve for web developers. Declarative UI Define the desired UI state, and React Native efficiently renders it on the screen. This simplifies development and improves code maintainability. Rich Ecosystem Abundant libraries and UI components accelerate development and enhance app functionality.

Node.js:

Server-Side Scripting Made Easy Build dynamic web servers using JavaScript, allowing for efficient server-side logic and communication with databases. Event-Driven Architecture Responds efficiently to user requests and data changes, making applications highly scalable and performant. Large Community and Ecosystem Benefits from a vast developer community and numerous libraries for various functionalities (databases, authentication, file systems). Working Together React Native Apps with Node.js Backends: A Node.js backend can handle authentication, database interactions, and complex calculations, offloading tasks from the mobile app for a smoother user experience. Real-time Communication Node.js can facilitate real-time communication features like push notifications and chat functionalities within React Native apps.

Data Sharing Node.js can act as an intermediary layer, managing data exchange between various services and the React Native app. The combination of React Native and Node.js empowers developers to create feature-rich mobile applications with efficient and scalable backends, all within a familiar JavaScript environment.

3.5 MONGO DB

Unveiling MongoDB A Flexible and Scalable NoSQL Database In the ever-evolving world of databases, MongoDB stands out as a powerful NoSQL document database. Unlike traditional relational databases with rigid schemas, MongoDB offers a flexible and scalable solution for storing and retrieving data. Here's a comprehensive dive into MongoDB's core concepts and its potential benefits for your project.

Core Concepts

- Document-Oriented Stores** data in flexible JSON-like documents, allowing for rich and diverse data structures within a single document. This approach is ideal for storing complex and evolving data models.
- Schema-less** Doesn't enforce a predefined schema for documents, providing greater flexibility in how data is organized. This is particularly beneficial for applications where data structures are constantly changing.
- Collections** Groups of related documents. Think of them as tables in a relational database, but with a less rigid structure.
- Documents** Individual units of data within a collection, similar to rows in a relational database table. Documents can have nested structures, allowing for complex relationships within the data.
- Key-Value Pairs** Documents are composed of key-value pairs, similar to how dictionaries work. Keys act as unique identifiers, and values can be various data types (strings, numbers, arrays, embedded documents).

Benefits of MongoDB:

- Scalability** Easily scales horizontally by adding more servers to handle growing data volumes. This makes it ideal for applications with unpredictable data growth.
- Performance** Optimized for fast reads and writes, making it suitable for real-time applications requiring quick data access.
- Flexibility** Accommodates diverse data structures and evolving schemas, perfect for applications where data models are not fully defined upfront.
- Rich Query Language** Supports a powerful query language (MongoDB Query Language - MQL) that allows for complex data filtering and retrieval.
- Schema Agnostic** Enables developers to focus on application logic rather than rigid data modeling,

improving development speed and agility. Integrations Integrates well with various programming languages and frameworks, providing development flexibility.

Use Cases:

Real-time Applications Ideal for applications requiring real-time data access and updates, such as chat applications, social media platforms, and IoT (Internet of Things) data storage. Content Management Systems (CMS) Stores content in a flexible format, allowing for easy management of diverse content types like text, images, and videos. E-commerce Platforms Manages product information, customer data, and order details with ease, adapting to evolving product catalogs and promotions. Location-Based Services Stores geospatial data efficiently, enabling applications like ride-hailing services and location-based recommendations.

CHAPTER 4

SYSTEM DESIGN

Branding:

Logo: A simple and recognizable logo that reflects the app's purpose and brand identity. Consider incorporating imagery related to vehicles or assistance.

Color Palette: A limited set of primary and secondary colors used consistently throughout the app. Choose colors that are easy on the eyes, convey trust and reliability, and complement the logo.

Typography: A set of fonts for headings, body text, buttons, and error messages. Choose fonts that are readable, professional, and work well at various sizes.

UI Components:

Buttons Different styles for primary actions (e.g., requesting assistance), secondary actions (e.g., canceling), and destructive actions (e.g., logging out). Maintain consistent styling for each button type. **Forms** Standardized layouts for user input, including text fields, dropdowns, and radio buttons. Implement clear labeling and validation for user input. **Cards** Reusable components to display information like mechanic profiles, breakdown details, and repair estimates. Ensure content hierarchy is clear with consistent padding and margins. **Navigation** A consistent navigation structure across all screens, allowing users to easily find desired features. Consider a bottom navigation bar or a hamburger menu, depending on the complexity of the app. **Icons** A set of icons for common actions, locations (map markers), and mechanic specialties. Maintain a consistent style and ensure icons are clear and easily understandable. **Loading Indicators** Visual cues to inform users when data is loading (e.g., progress bars, spinners). Implement a consistent style and duration for loading animations.

Request Assistance:

A prominent "Request Assistance" button for easy access. User location automatically populated using GPS. Simple form to describe the breakdown issue with options for pre-defined problems

or a text input field. Ability to upload photos for visual reference. Estimated wait time based on mechanic availability in the area.

Mechanic Listing:

List of available mechanics in the vicinity displayed on a map or list view. Information cards for each mechanic including profile picture, name, rating, average response time, and area of expertise. Option to filter mechanics based on rating, expertise, or distance.

Mechanic App:

Available Requests. A dedicated screen displaying nearby breakdown requests in real-time. Requests displayed on a map or list view, highlighting the user location and breakdown details. Option to filter requests based on location, vehicle type, or issue type.

Accepting Requests:

Clear "Accept" or "Decline" button associated with each request. Ability to view user profile and vehicle information before accepting a request. Option to communicate with the user via in-app chat to clarify the issue or provide an estimated arrival time.

Navigation:

Utilize map integration to navigate efficiently to the user's location. Estimated arrival time displayed for the user within the chat.

Service Completion:

Option to mark the service as completed within the app. Ability to document repairs performed or suggest repair options. Secure payment processing through the app for repair costs. Option to leave feedback for the user after service completion.

Interaction Design:

Animations Subtle animations to enhance user experience for actions like button clicks, screen transitions, and loading states. Ensure animations are smooth and don't distract users. Microinteractions Small interactions that provide feedback to users (e.g., button press animations,

confirmations). These enhance usability and keep users engaged. Error Handling Clear and informative error messages that guide users when encountering issues. Use concise language and offer solutions or retry options.

Accessibility:

Color Contrast Ensure sufficient color contrast between text and background elements for users with visual impairments. Font Size Maintain a readable font size across all UI elements.

Text Alternatives Provide alternative text descriptions for images and icons to assist users with screen readers.

4.1 SYSTEM ARCHITECTURE

The Ride Right Breakdown Assistant system relies on a robust three-tier architecture to ensure efficient service delivery, data security, and scalability. Here's a breakdown of each layer:

User-Facing Tier (Mobile Apps):

User App Drivers interact with the system through a user-friendly mobile app. Features include requesting assistance, locating nearby mechanics, in-app chat, and secure payment processing.

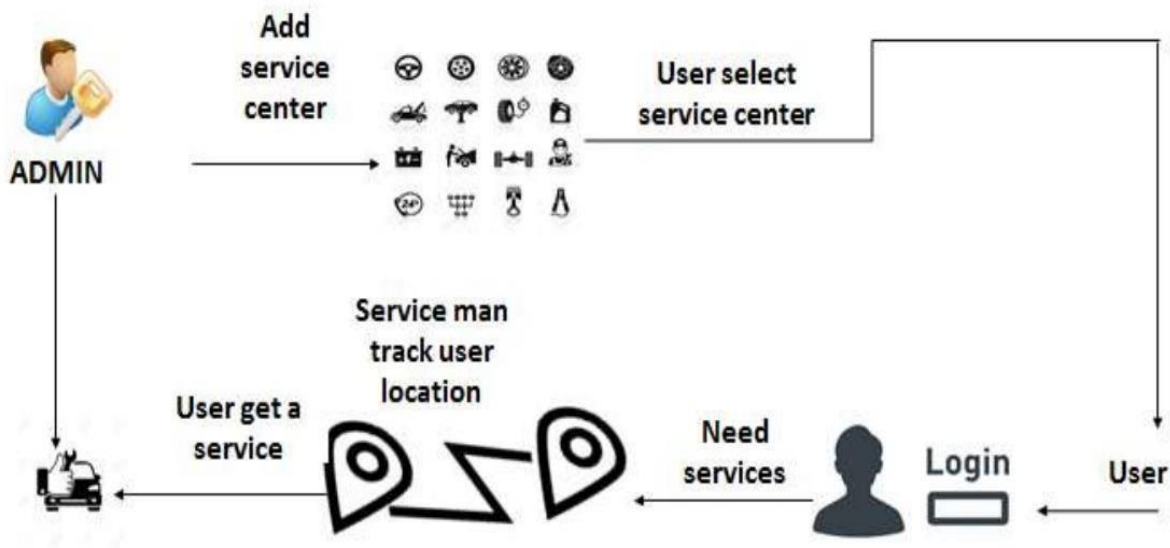
Mechanic App Mechanics utilize a dedicated app to view available requests, accept jobs, navigate to user locations, and manage service completion. Both apps share a consistent design language for a unified user experience.

Secure Backend (Cloud-Based):

Server A scalable server running on a cloud platform (e.g., Google Cloud Platform, Amazon Web Services) handles all backend operations. This ensures high availability and the ability to handle increasing user demand.

Database (MongoDB) A NoSQL database like MongoDB stores user data, mechanic profiles, breakdown requests, and potentially historical data for analytics. This flexible schema allows for diverse data structures and easy adaptation to evolving needs.

API Gateway An API gateway acts as a single entry point for all communication between the mobile apps and the backend server. It ensures secure access and manages incoming requests efficiently.



Central Server:

API Gateway: Acts a single entry point for all communication between the mobile apps (User and Mechanic) and the backend server. It routes requests securely and manages traffic flow.

Cloud Server: The core server infrastructure, likely hosted on a cloud platform (e.g., Google Cloud Platform or Amazon Web Services), that handles data storage, processing, and business logic.

Database: Stores all system data, including user information, mechanic profiles, breakdown requests, and potentially historical data for analytics. The diagram suggests a relational database (e.g., MySQL, PostgreSQL) but doesn't specify the exact type.

Data Flow:

User Requests Assistance: The user initiates a request for assistance through the User App, providing details about the breakdown.

Request to Server: The User App transmits the request data securely through the API Gateway to the server.

Server Processes Request: The server processes the request, potentially using the user's location to find nearby mechanics.

Mechanic Availability: The server retrieves information about available mechanics in the user's vicinity from the database.

Request Sent to Mechanics: The server sends the breakdown request to relevant mechanics through the API Gateway.

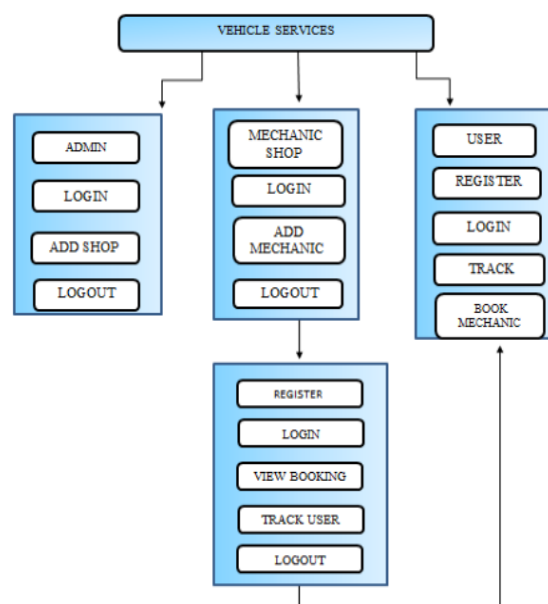
Mechanic Accepts Request: A mechanic views the request details on the Mechanic App and accepts the job.

Confirmation to User: The server confirms the request acceptance with the user through the User App and provides mechanic details (e.g., estimated arrival time).

Mechanic Navigates: The mechanic utilizes the built-in navigation service within the Mechanic App to reach the user's location.

Service Completion: The mechanic performs the necessary repairs, and the user confirms service completion through the User App (potentially after payment processing).

Admin Manages System: The Admin Panel allows administrators to oversee user registrations, mechanic profiles, manage service fees, analyze system performance metrics, and update system content.



4.2 DFD DIAGRAM

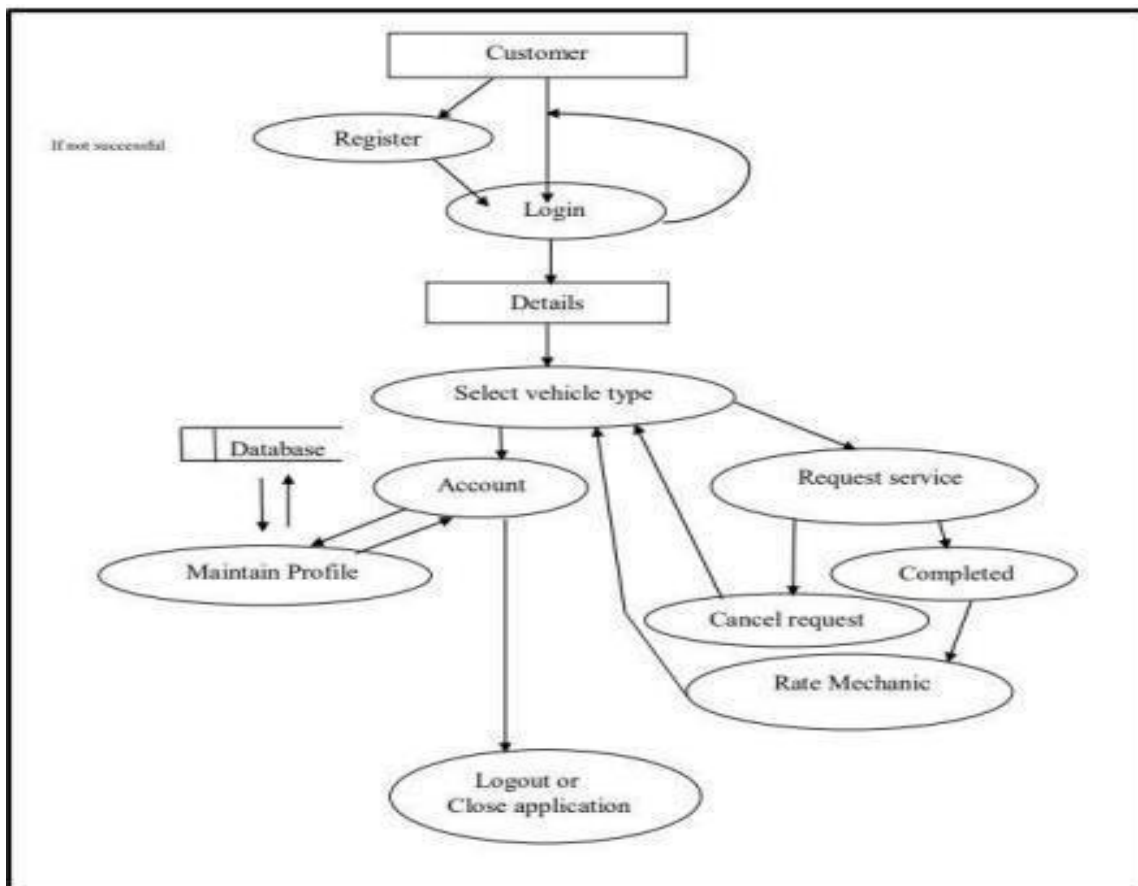
Creating a full Data Flow Diagram (DFD) for a Ride Right breakdown system would typically involve several levels of detail and components. However, I can provide a simplified DFD for a high-level overview of the system. In this DFD, we'll focus on the major processes and data flows:

Level 0 DFD (Context Diagram):

User: Represents the driver using the mobile application to request assistance in case of a vehicle breakdown.

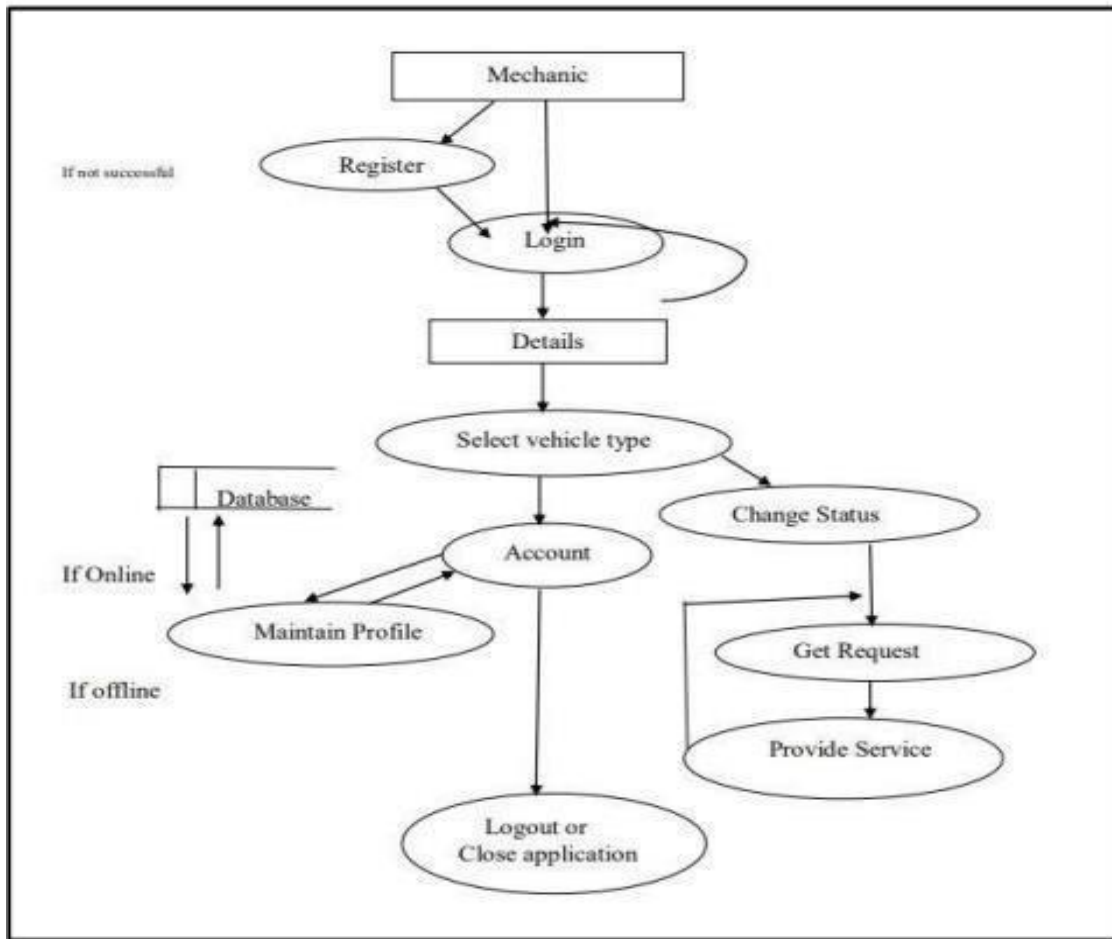
Mechanic: Represents a qualified mechanic using their mobile application to view and accept breakdown requests.

Admin: Represents the system administrator who manages user and mechanic registrations, sets service fees, and monitors system performance through a web interface.



Level 1 DFD:

At this level, The flowchart starts with the mechanic being "Online" within the mechanic mobile application, signifying their availability to accept service requests. New Request When a new request comes in, the mechanic receives a notification through the app. Request Details The mechanic can then view the details of the request, which likely include User location on a map, User information, Breakdown details ,Estimated distance to the user, Accept or Decline Based on the request details, the mechanic can choose to Accept If the mechanic decides to accept the request, they proceed to the next step. Decline If the mechanic chooses to decline, the reason for declining (optional) might be logged, and the request is likely offered to other available mechanics. Navigation Once the request is accepted, the mechanic app provides turn-by-turn navigation to the user's location, using an integrated mapping service. Service Completion Upon reaching the user's location, the mechanic performs the necessary repairs or services. User Confirms Completion The user confirms service completion through the user mobile app (potentially after payment processing). Feedback The mechanic might have the option to leave feedback about the user's experience. This data might be used for internal quality control purposes. End The flowchart concludes with the mechanic returning to an "Online" state, ready to receive new requests.



This Level 1 DFD provides an overview of this flowchart and offers a simplified illustration of the mechanic's experience within the on-road vehicle breakdown assistant system, focusing on how they receive requests, navigate to user locations, and complete service calls.

4.3 ER DIAGRAM

An Entity-Relationship Diagram (ERD) visually depicts the entities (data objects) in a system and the relationships between them. Here's a breakdown of a potential ERD for the Ride Right Vehicle Breakdown Assistant system

Entities:

User:

Represents a registered user of the application who can request assistance in case of a vehicle breakdown. Attributes might include:

- User ID (Primary Key)
- Name
- Phone Number
- Email Address
- Vehicle Information (optional) - Make, Model, Year

Mechanic:

Represents a qualified mechanic registered with the system who can accept breakdown requests and provide service. Attributes might include

- Mechanic ID (Primary Key)
- Name
- Contact Information (Phone, Email)
- Area of Expertise (e.g., Engine Repair, Electrical Systems)
- Rating (Average user rating)

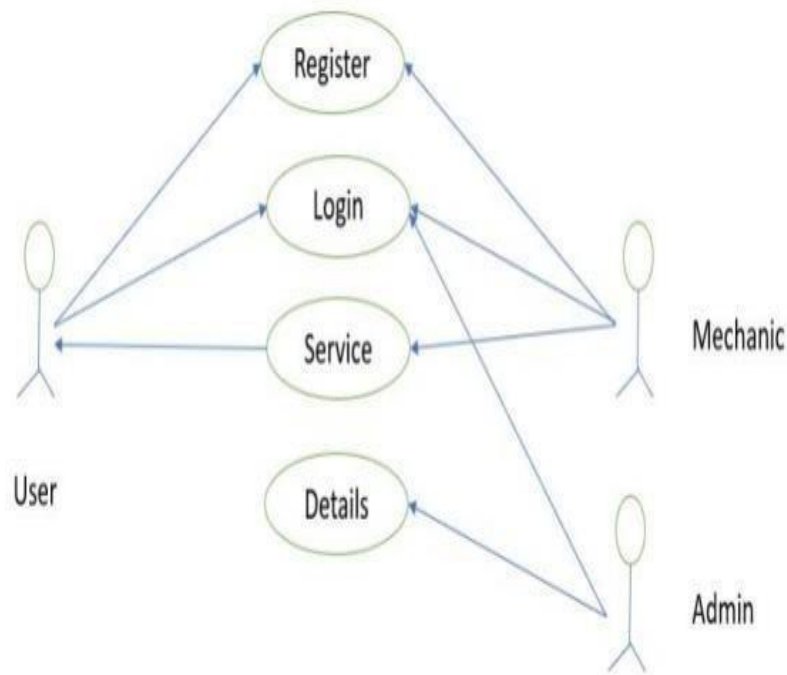
Availability Status (Online/Offline)

Breakdown Request:

Represents a specific request for assistance submitted by a user. Attributes might include Request ID (Primary Key), User ID (Foreign Key referencing User), Mechanic ID (Foreign Key referencing Mechanic - initially null, assigned later), Location (User's GPS coordinates), Breakdown Details (Description of the problem), Status (Open, Accepted, In Progress, Completed) Timestamp (Date and Time of request).

Payment:

Represents a financial transaction for services rendered. Attributes might include Payment ID (Primary Key), Breakdown Request ID (Foreign Key referencing Breakdown Request) Amount, Payment Method (e.g., Credit Card, Cash), Timestamp (Date and Time of payment)



Entities Boxes represent entities, which are real-world aspects the system stores data about. Here, the entities are User, Mechanic, Breakdown Request, and Service History. Attributes Ovals within the boxes represent attributes, which are specific data points about an entity. Examples include User Name, Mechanic Expertise, and Request Status. Relationships Lines connect the entities, signifying relationships between them. Diamond shapes on the lines indicate the cardinality (how many of one entity relate to another). The user can submit many Breakdown Requests, each assigned to one Mechanic. A Mechanic can fulfill many Requests, and each Request results in zero or one Service History record.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 MODULE DESCRIPTION

The user module plays a critical role in the Ride Right breakdown assistance system, offering a streamlined experience for drivers seeking roadside help. This section explores the functionalities, features, and potential benefits of this module.

User module:

Registration:

Users create accounts by providing essential details Name, Email address, Phone number, Secure password creation, Optionally, they can add vehicle information Vehicle type (car, motorcycle, etc.) Make, Model, Year ,Secure password management practices like encryption and two-factor authentication are essential to protect user data.

Login:

Returning users log in with their registered email address and password. The system facilitates a secure login process with features like password masking and a forgot password reset mechanism.

Request Assistance:

This central function offers a quick and efficient way to request roadside assistance. Key features include Location Services Integration Using GPS, users can automatically share their current location upon initiating a request, eliminating manual address entry. Manual Location Entry Alternatively, users can enter their breakdown location manually if GPS is unavailable or inaccurate. Breakdown Description Users provide a brief description of the problem (e.g., flat tire, engine overheating, electrical issues). Vehicle Selection If vehicle information wasn't provided during registration, users can select their vehicle type here. This helps mechanics assess the situation better. Emergency Contact Information Users can choose to share emergency contact details to be notified alongside them in case of a serious issue.

View Available Mechanics:

When a user requests assistance, the system identifies nearby available mechanics. Users can view a list or map of these mechanic profiles with pictures and ratings (if available) Estimated arrival times based on distance and traffic conditions Option to filter by expertise

Choose Mechanic:

Users can choose a preferred mechanic based on Ratings and reviews, Expertise , Estimated arrival time

Track Mechanic Location:

Once a request is accepted by a mechanic, users can track their location in real-time on a map. This provides transparency and peace of mind by showing when the mechanic is on their way.

Mechanic module:

Name, Email address, Phone number, Secure password creation, Expertise areas (engine repair, electrical work, etc.) Business details This can include business name, logo, service offerings, and contact information. Uploading relevant documents Licenses, certifications, or insurance documents can enhance user trust and credibility.

Login:

Existing mechanics log in using their registered email address and password. Similar to the user module, secure login practices like password masking and a forgot password reset mechanism ensure secure access.

Set Availability:

This crucial feature allows mechanics to manage their working hours and workload. Mechanics can switch between "online" and "offline" statuses, indicating their availability to accept service requests.

View Requests:

When a user initiates a request for assistance in a mechanic's vicinity, it appears in their "Requests" section. Each request displays key information, Breakdown location Shown on a map for efficient navigation. Breakdown description Briefly outlining the user's reported problem. User details Contact information for potential communication. Vehicle type (if provided by the user)

Accept/Decline Requests:

Mechanics can review request details and choose to Accept the request, indicating their willingness to service the user's vehicle. Decline the request, potentially due to unavailability, expertise mismatch, or other reasons.

In-App Chat:

This feature facilitates communication between mechanics and users within the app.

Mechanics can use the chat to:

Clarify breakdown details and ensure they have the necessary tools for the job.

Inform users of estimated arrival times.

Provide updates on their progress.

Manage Services:

This section allows mechanics to manage accepted service requests. It might include functionalities like Viewing service details and breakdown descriptions. Updating service status (e.g., "traveling to user location," "at user location," "service completed"). Recording service details (e.g., parts used, repairs performed) for future reference and billing purposes (optional).

Business Management:

Mechanics with established businesses can manage their business profiles within the app. This might include Updating business name, logo, and contact information. Managing service offerings and potentially setting service fees.

View Feedback:

Mechanics can access user ratings and reviews left after service completion.

This feedback offers valuable insights into user experience and allows mechanics to identify areas for improvement.

Earnings Management:

This feature (integrated with a payment gateway) allows mechanics to view their earnings history within the system. Mechanics can track payments received for completed services.

Settings:

Similar to the user module, the mechanic module provides access to settings like Frequently Asked Questions (FAQs), Help guides and tutorials, System settings and user guides, Log out functionality

Admin module:

The admin module forms the backbone of the Ride Right breakdown assistant system. It empowers system administrators to oversee various aspects, ensuring smooth operation and a positive user experience. Here's a breakdown of the functionalities and potential benefits of the admin module:

User Management:

User Registration Review & Approval In systems with moderation enabled, admins can review and approve/reject user registrations. This helps maintain quality control and prevent fraudulent accounts.
User Details & Activity History: Admins can view detailed information about registered users, including Name, Email address, Phone Number Vehicle information (if provided), Registration date, Past service request history ,
User Feedback & Reports Management Admins can access and manage user feedback, Reviews left for mechanics General feedback about the system functionality Analyze reports to identify areas for improvement and enhance user satisfaction.

Mechanic Management:

Mechanic Application Review & Approval: Admins can review applications submitted by mechanics seeking to join the platform. This review process can involve Verifying qualifications and licenses, Checking references, Assessing areas of expertise, **Mechanic Profile & Service History View** Admins can access detailed profiles of registered mechanics, including Name and contact information, Expertise areas, Ratings and reviews from users, Service history, including completion rates and user feedback.
Service Fee Guidelines Management Admins can set baseline

service fee guidelines for different types of repairs or services. Mechanics can set their own fees within these guidelines, providing flexibility for users.

System Management:

System Performance Monitoring Admins can monitor the overall performance of the system, including:

- User activity
- Mechanic availability
- Request processing times
- System uptime and response rates

Content Management Admins can manage content displayed within the app for both users and mechanics, including:

- Frequently Asked Questions (FAQs)
- Help guides and tutorials
- Announcements or system updates

System Configuration & Settings Admins can configure various system settings, such as:

- Location services settings
- Push notification settings
- User registration moderation settings (optional)

Reporting & Analytics:

Admins can generate comprehensive reports on various aspects of the system:

- User registration trends
- Mechanic activity and service completion rates
- User feedback and service ratings
- Breakdown frequency and types (optional)
- Analyzing these reports provides valuable insights to improve service offerings, identify potential issues, and optimize platform functionality.

Benefits of Admin Modules:

Quality Control: By reviewing user registrations and mechanic applications, admins can maintain a high standard of service providers and ensure user safety. Data-Driven Decision Making System performance monitoring and reporting offer valuable data to make informed decisions about platform features, service fees, and future development. Improved User Experience Admins can leverage user feedback and reports to identify areas for improvement and enhance the overall user experience. Efficient System Management The ability to monitor system performance, manage content, and configure settings allows admins to maintain a smooth-running platform. Transparency & Trust Admins can ensure transparency by providing clear guidelines and procedures for users and mechanics.

5.2 SAMPLE CODE

Creating a complete sample code for a Ride Right Breakdown Assistant System is beyond the scope of a simple response due to the complexity and various technologies involved. However, I can provide you with a simplified java script code snippet to give you a basic understanding of how user and mechanic profiles might be managed within such a system.

Here's a minimal example of how you can create user and mechanic profiles using Java Script:

```
// Registration
```

```
function registerUser(userData) {
```

```
    // Create a new user object
```

```
    const user = { ...userData };
```

```
    // Perform validation on user input
```

```
    if (!validateUserData(user)) {
```

```
        return { error: "Invalid data" };
```

```
    }
```

```
    // Encrypt password before storing
```

```
    user.password = encryptPassword(user.password);
```

```
    // Save user data to database (using a suitable database library/API)
```

```

    return saveUserToDatabase(user);

}

// Login

function loginUser(email, password) {

    // Fetch user from database by email

    const user = fetchUserFromDatabase(email);

    if (!user) {

        return { error: "User not found" };

    }

    // Compare provided password with encrypted password

    if (comparePasswords(password, user.password)) {

        return user; // Successful login

    } else {

        return { error: "Invalid password" };

    }

}

// Request assistance

function requestAssistance(userID, breakdownDetails, location) {

    // Create a new breakdown request object

    const request = {

        userID,

        breakdownDetails,

        location,

```

```

    status: "pending",

    // ...other request details

};

// Send request to server for processing (using a suitable API)

return sendRequestToServer(request);

}

// Track mechanic location

function trackMechanicLocation(requestID) {

    // Fetch mechanic's real-time location from server (using a suitable API)

    const mechanicLocation = fetchMechanicLocation(requestID);

    // Display mechanic's location on a map (using a map library like Google
    Maps)

    displayMechanicLocationOnMap(mechanicLocation);

}

// ... (other user module functions)

// Set availability

function setAvailability(mechanicID, isAvailable) {

    // Update mechanic's availability status in database

    return updateMechanicAvailability(mechanicID, isAvailable);

}

```

CHAPTER 6

SYSTEM TESTING

6.1 METHODOLOGY

The methodology Developing an Ride Right breakdown assistant system involves various stages and considerations. Here's a breakdown of the key methodologies to follow:

Requirement Gathering & Analysis: Identify target users: Understand the needs of both drivers (user) and roadside assistance providers (mechanics). Define functionalities: Outline the core features users and mechanics require, including request assistance, mechanic search, location tracking, feedback, and potentially payment processing. Prioritize features: Determine the most essential functionalities to be implemented initially and plan for future enhancements.

System Design: System Architecture: Design the overall architecture of the system, including the user interface (mobile app), backend server, and database. Data Modeling: Design a data model to store user information, mechanic profiles, breakdown requests, service details, ratings, and reviews. If the system integrates with external services like mapping APIs or payment gateways, define APIs for communication.

Technology Stack Selection: Front-End Development: Choose a framework for developing the user interface (e.g., React Native, Flutter) for cross-platform mobile app compatibility. Back-End Development: Select a technology stack for the server-side (e.g., Node.js, Python) to handle request processing, data storage, and API communication. Database: Choose a suitable database technology (e.g., MySQL, PostgreSQL) to store user, mechanic, and request data securely.

Development & Implementation: Develop user interface: Create a user-friendly and intuitive mobile app interface for users to request assistance, search for mechanics, track location, and provide feedback. Develop server-side logic: Implement functionalities on the backend server for request processing, mechanic matching, location tracking, and communication between users and

mechanics. Integrations (Optional): Integrate with third-party services like mapping APIs (e.g., Google Maps) for location services and potentially payment gateways for secure transactions.

Testing & Deployment: Unit testing: Perform thorough unit testing to ensure individual modules function as expected. Integration testing Test the interaction between different parts of the system. Usability testing Conduct usability testing with target users to identify and address any user interface or experience issues. Deployment Deploy the application to a suitable platform (e.g., app stores) for user access.

Maintenance & Updates: Monitor system performance: Continuously monitor system performance and resource usage. Address bugs and issues: Respond promptly to identified bugs and address user feedback to improve the application. Implement new features: Based on user needs and market trends, develop and integrate new features to enhance the system's functionality and user experience. Security Implement robust security measures throughout the development process. This includes secure user authentication, data encryption, and protection against potential vulnerabilities. Scalability Design the system to be scalable to accommodate a growing user base and increased number of breakdown requests. Compliance Adhere to any relevant regulations related to user data privacy and security. Following this methodology will help create a robust, user-friendly, and dependable on-road vehicle breakdown assistant system that caters to the needs of both drivers and service providers.

CHAPTER 7

CONCLUSION

The Ride Right breakdown assistant system offers a valuable solution for drivers who experience unexpected car troubles. By leveraging technology and connecting users with reliable mechanics, the system provides Peace of mind Users gain peace of mind knowing they can easily request assistance in case of a breakdown, anytime and anywhere. Efficiency The system streamlines the process of finding a mechanic, reducing wait times and stress during a stressful situation.

Transparency Users can view available mechanics, estimated arrival times, and track the mechanic's location in real-time. Quality control Admins can ensure a high standard of service by reviewing mechanic applications and user feedback. Data-driven insights Analyzing user activity and feedback helps in improving the platform and service offerings. The system benefits not only users but also mechanics Increased business opportunities Mechanics gain access to a wider customer base through the app. Improved efficiency Managing requests and scheduling becomes more efficient with the app's functionalities. User feedback Mechanics can receive valuable feedback to improve their services. Looking forward Integration with payment gateways can allow for secure in-app transactions. Advanced features like towing services or parts ordering can be explored for a more comprehensive solution. Implementing a machine learning model could predict potential breakdowns based on user data and driving patterns, offering preventive maintenance suggestions. By continuously innovating and adding features, the on-road vehicle breakdown assistant system can become an indispensable tool for both drivers and mechanics, ensuring a smoother and safer on-road experience for everyone.

CHAPTER 8

FUTURE ENHANCEMENT AND FEATURES

Advanced Vehicle Diagnostics:

Integrate with a user's vehicle's On-Board Diagnostics (OBD) port to potentially retrieve real-time data about the breakdown issue. This can help mechanics remotely diagnose problems and provide initial assistance.

Trip Planning & Alerts:

Integrate with navigation apps to suggest nearby mechanics and service stations along planned routes. The system could also send proactive alerts for upcoming maintenance needs based on vehicle data (optional).

Real-Time ETA Updates:

Utilize advanced traffic prediction algorithms to provide more accurate estimated arrival times for mechanics, reducing user wait times and anxiety.

Payment Integration: Integrate secure in-app payment options to allow users to seamlessly pay for services directly through the app.

Advanced Towing & Recovery:

Partner with towing and recovery services to offer integrated solutions within the app, simplifying the process for users in need.

Community Features:

Implement a user community forum or chat feature where users can share experiences, ask questions, and recommend mechanics.

Improved Mechanic Features:

Skill Verification & Specialization:

Implement a system for verifying mechanics' skills and areas of specialization beyond self-reported expertise. This can enhance user confidence when choosing a mechanic.

Inventory Management:

Allow mechanics to manage their parts inventory within the app, enabling them to check stock availability before accepting requests.

Job Bidding System:

Consider a system where users can post breakdown details and mechanics can bid for the job based on expertise and estimated repair costs. This can increase competition and potentially provide users with more affordable service options.

Performance Tracking & Incentives: Implement a system to track mechanic performance based on user ratings, response times, and completion rates. Offer incentives for high-performing mechanics to encourage quality service.

Advanced System Features:

AI-powered Chatbots:

Integrate AI-powered chatbots to provide basic troubleshooting assistance to users and answer common user questions, reducing reliance on customer support.

Predictive Maintenance:

Analyze user data and vehicle information to predict potential breakdowns and suggest preventive maintenance schedules.

Fleet Management Integration (Optional):

Partner with fleet management companies to offer breakdown assistance solutions specifically tailored to fleet vehicles, potentially including real-time tracking and maintenance management tools.

CHAPTER 9

APPENDIX

9.1 CODING

```
<! import React, {useState, useEffect} from 'react';

import {

  StyleSheet,

  Text,

  View,

  TextInput,

  TouchableOpacity,

  Dimensions,

  Image,

  BackHandler,

  PermissionsAndroid,

  Alert

} from 'react-native';

import axios from 'axios';

import { Icon, Button } from 'react-native-elements';

import RNRestart from 'react-native-restart'; // Import package from node modules

import LocationView from 'react-native-location-view';

import Icon2 from 'react-native-vector-icons/FontAwesome5';

import AsyncStorage from '@react-native-async-storage/async-storage';
```

```

import GetLocation from 'react-native-get-location';

export default function App(props) {

  const [destination, setdestination] = useState({

    latitude: 12.7898,

    longitude: 80.2542,

  });

  const [currentLongitude, setCurrentLongitude] = useState("");

  const [currentLatitude, setCurrentLatitude] = useState("");

  const [sosActivated, setbrakeActivated] = useState(false);

  const handlebrakePress = () => {

    // Implement your brake functionality here.

    // For demonstration purposes, we'll just log a message.


    AsyncStorage.getItem('ipaddress').then((ipaddress) => {

      const data = {

        "device_id": "9150421916",

        "lat": currentLatitude,

        "lon": currentLongitude,

        "status": true

```

```
};
```

```
useEffect(() => {
```

```
  const requestLocationPermission = async () => {
```

```
    try {
```

```
      const granted = await PermissionsAndroid.request(
```

```
        PermissionsAndroid.PERMISSIONS.ACCESS_FINE_LOCATION,
```

```
        {
```

```
          title: 'Location Access Required',
```

```
          message: 'This App needs to Access your location',
```

```
        },
```

```
      );
```

```
      if (granted === PermissionsAndroid.RESULTS.GRANTED) {
```

```
        //To Check, If Permission is granted
```

```
        getOneTimeLocation();
```

```
      } else {
```

```
      }
```

```
    } catch (err) {
```

```
      console.warn(err);
```

```
    }
```

```
  };
```

```
  requestLocationPermission();
```

```

return () => {
  size={150}
  color={sosActivated ? 'red' : 'black'}
  />
</TouchableOpacity>
<Button
  title="Request for help"
  onPress={handlebrakePress}
  buttonStyle={styles.sosButton}
  />
</View>

</View>
);
}
}

```

```

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#000000',
    alignItems: 'center',
    justifyContent: 'center',
  },

```

```
logo: {  
  fontWeight: 'bold',  
  fontSize: 35,  
  color: '#DA70D6',  
  marginBottom: 40,  
},  
inputView: {  
  width: '80%',  
  backgroundColor: '#465881',  
  borderRadius: 25,  
  height: 50,  
  marginBottom: 20,  
  justifyContent: 'center',  
  padding: 20,  
},  
inputText: {  
  height: 50,  
  color: 'white',  
},  
forgot: {  
  color: 'white',  
  fontSize: 11,  
},  
loginBtn: {
```

```
width: '80%',

backgroundColor: '#DA70D6',

borderRadius: 25,

height: 50,

alignItems: 'center',

justifyContent: 'center',

marginTop: 40,

marginBottom: 10,

},

container: {

  flex: 1,

  justifyContent: 'center',

  alignItems: 'center',

},

sosButton: {

  marginTop: 20,

  backgroundColor: 'red', // Customize button color

},

loginText: {

  color: 'white',

},

});
```

9.2 SCREENSHOTS



The screenshot shows a login interface with a dark blue background. At the top, the text "Vehicle Breakdown" is displayed in a light blue, italicized font. Below it, the word "LOGIN" is centered in a white, bold, sans-serif font. There are three white input fields stacked vertically. The first field is labeled "Enter Username" in a light blue, italicized font. The second field is labeled "Enter Password" in the same font style. The third field is labeled "IP Address" in the same font style. Below the input fields is a large, rounded orange button with the word "LOGIN" in a white, bold, sans-serif font.

9.1.1 User Login



A login form titled "MECHANIC LOGIN" on a dark blue background. It features three white input fields for "User Id", "Password", and "IP Address", followed by an orange "LOGIN" button.

MECHANIC LOGIN

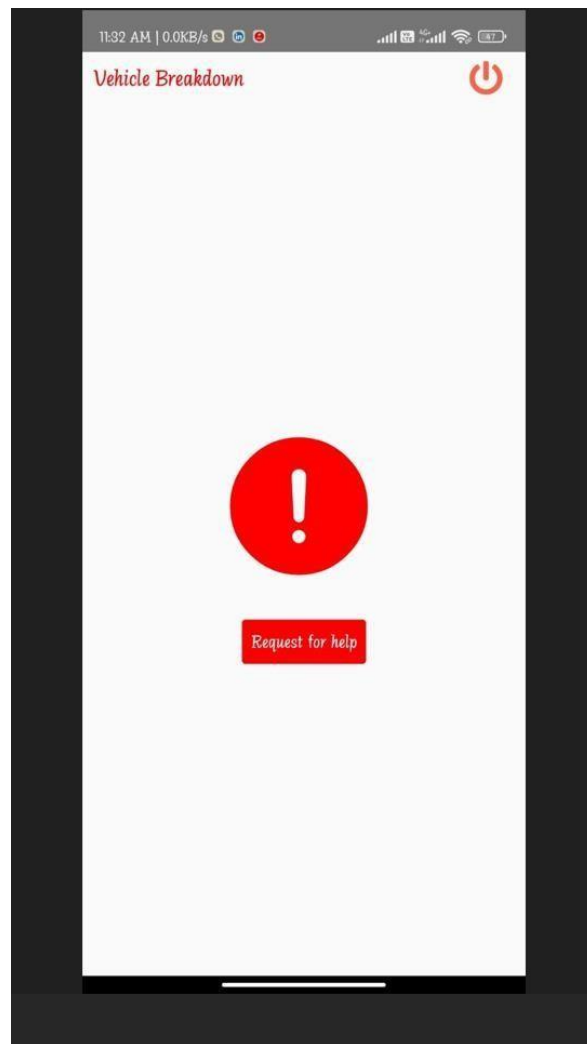
User Id

Password

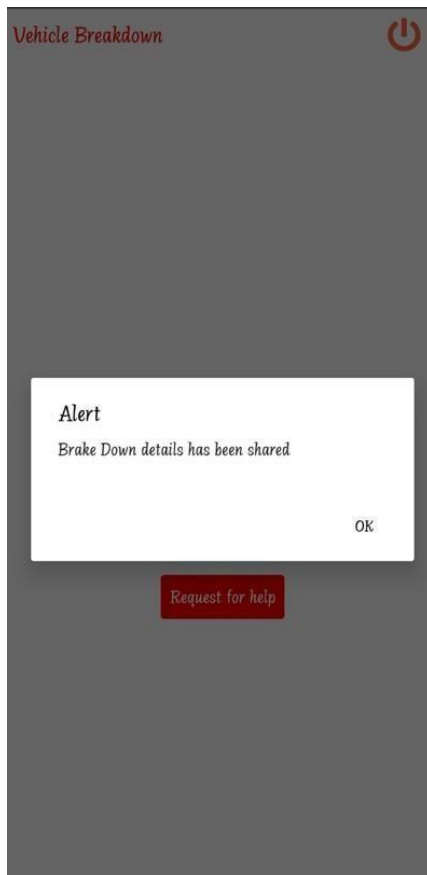
IP Address

LOGIN

9.1.2 Mechanic Login



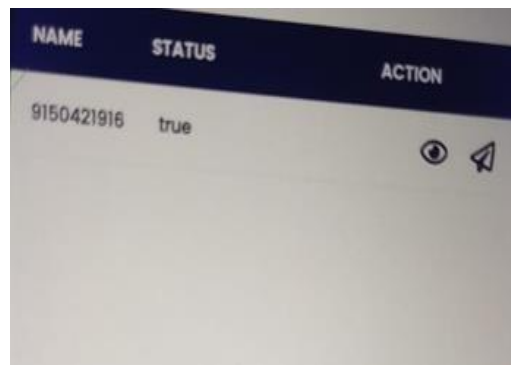
9.1.3 User Request



9.1.4 Alert Message



9.1.5Mechanic home



9.1.6 Request Status

CHAPTER 10

REFERENCE

- [1] L. Awange, “The global positioning system,” in Environmental Science and Engineering (Subseries: Environmental Science), 2012.
- [2] M. Purvis, J. Sambells, and C. Turner, Google Maps Applications with PHP and Ajax From Novice to Professional. 2006.
- [3] A. Ankolekar et al., “DAML-S: Web Service description for the semantic Web,” in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2002, doi: 10.1007/3-540-48005-6_27.
- [4] D. Hughey, “The Traditional Waterfall Approach,” University of Missouri. 2017.
- [5] N.d., “SDLC Waterfall Model,” Tutorialspoint, 2013.
- [6] N. B. Ruparelia, “Software development lifecycle models,” ACM SIGSOFT Softw. Eng. Notes, 2010, doi: 10.1145/1764810.1764814.
- [7] M. McCormick, “Waterfall and Agile Methodology,” MPCS Inc, 2012.
- [8] H. Barry, “What is Waterfall model- advantages, disadvantages and when to use it?,” Istqb Exam Certification, 2012. .
- [9] R. Pressman, “Software Engineering Seventh Edition,” Metod. waterfall, 2015.
- [10] N. A. Govoni, “Flowchart,” in Dictionary of Marketing Communications, 2012.
- [11] R. J. Wieringa, “Entity-Relationship Diagrams,” in Design Methods for Reactive Systems,

2003.

[12] S. Burge, “The Systems Engineering Tool Box,” Chart, 2011.

[13] R. J. Wieringa, “Data Flow Diagrams,” in Design Methods for Reactive Systems, 2003.

[14] V. Mauger, “Interface,” in The Routledge Companion to Video Game Studies, 2014.

[15] R. Saltini and R. Akkerman, “Testing improvements in the chocolate traceability system:

Impact on product recalls and production efficiency,” Food Control, 2012, doi:

10.1016/j.foodcont.2011.07.015

[16] Bo Pang., and Lillian Lee. (2004)’. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In Proceedings of the 42nd annual meeting on Association for Computational Linguistics, (page 271).

[17] Ziebinski A., Cupek R., Grzechca D., and chruszczyk L. (2017)’. Review of advance driver assistance system (ADAS) (DOI: 10.1063/1.5012394).

[18] Singh Puneet, Kapoor Ashutosh, Kaushik Vishal, and Bindu Hima. (2010)’. Architecture for automated tagging and clustering of song _les according to mood. International Journal of Computer Science Issues (IJCSI), 7(4).

[19] Jin Wei. (2008)’. Mining hidden associations in text corpora through concept chain and graph queries. ProQuest,

[20] Thayer Robert. (1989)’. The biopsychology of mood and arousal. Oxford University Press.

[21] Neviarouskaya A., Prendinger H., and Ishizuka, M (2009)’. Semantically distinct verb classes involved in sentiment analysis. In: Proceedings of the International Conference on Applied Computing (AC 2009), Japan, (pp. 27–34).

- [22]” Haridas K, Ahmad S. Baharudin, Kamal & karkonasasi (2016)’. Automotive Servicing and Breakdown Assistance System (ASBAS): Impact of Perceived Ease of Use (PEOU)
- [23]. D. Selvathi, P. Pavithra and T. Preethi, "Intelligent transportation system for accident prevention and detection, " in 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), India, 2017.
- [24]. WHO, "WORLD HEALTH ORGANISATION, "2018. Online]. Available: http://www.who.int/violence_injury_prevention/road_traffic/en/. Accessed 28 July 2018].
- [25]. V. Ahmed and N. P. Jawarkar, "Design of LowCost Versatile Microcontroller Based System Using Cell Phone for Accident Detection and Prevention, " in 2013 6th International.
- [26]. Indranil Nikose, Tushar Raut, "Review Paper on Smart Helmet using GSM and GPS Technology", International Journal of Advanced Research in computer and communication engineering, vol.6. Issue 2, February 2017.
- [27]. Nitin Agarwal, Anshul Kumar Singe, "Smart Helmet", International Research Journal of Engineering and Technology", Volume 02 Issue: 02 , May-2015 .