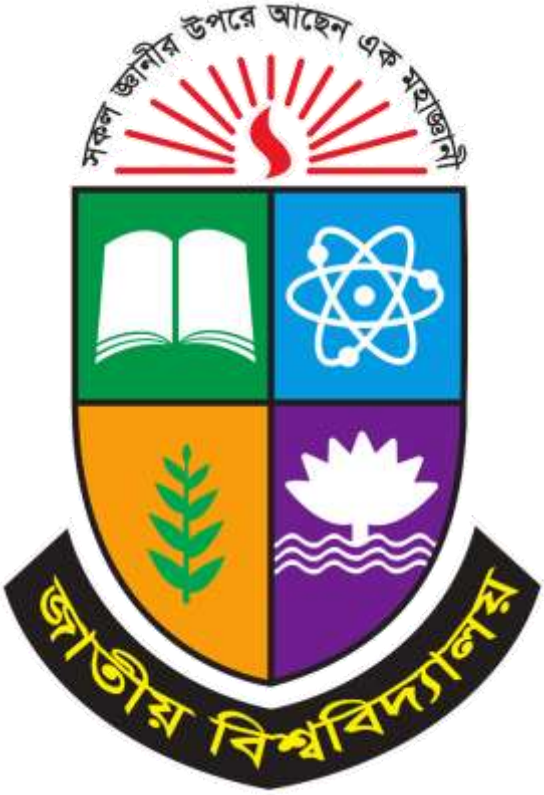
On Road Vehicle Breakdown Service in Bangladesh

By

Shanto kumar Das

&

Mahjabin Sultana



Project for the Degree of Bachelor of Science

In

Computer Science & Engineering

NATIONAL UNIVERSITY 2022

Project for the Degree of Bachelor of Science

On Road Vehicle Breakdown System

Submitted By:

Shanto Kumar Das

Registration Number:18502004092

&

Mahjabin Sultana

Registration Number:18502004098

Supervised By:

DIPTI SUTRODHOR

LECTURER

Department of CSE(UIBT)

Submitted to the Department of Computer Science and Engineering of Uttara Institute of Business and Technology (UIBT)

In partial fulfilment of the requirements for the degree of Bachelor of Science

Project Report Approval

Shanto Kumar Das

Registration Number:18502004092

Mahjabin Sultana

Registration Number:18502004098

Project Title: On Road Vehicle Breakdown Service in Bangladesh

We the undersigned, recommend that the project completed by the student listed above, in partial fulfillment of Bachelor of science requirements, be accepted by the Department of Computer Science & Engineering, Uttara Institute of Business and Technology (UIBT) for deposit.

Supervisor Approval

DIPTI SUTRODHOR

Department of CSE(UIBT)

Departmental Approval

DIPTI SUTRODHOR

LECTURER

ACKNOWLEDGEMENT

In completing this eService project, I have been fortunate to have help, support, and encouragement from many people. I would like to thank them for their cooperation. I am sincerely thankful to Dipti Sutrodhor ma'am, for his wonderful supervision during this eService project work in the field of Computer Science, at UIBT. I am also thankful for his continuous feedback and encouragement throughout this project work. His broad knowledge and hardworking attitude have left me with very deep and cool impressions, make me energetic and will greatly benefit me throughout my life. Last but not the least; Above all, we should not forget the great director of the world, 'The Almighty'; let us thank the Almighty for His inspiration.

ABSTRACT

This project focuses on the development of an "**On Road Vehicle Breakdown Service**" system aimed at providing timely assistance to drivers facing unexpected vehicle malfunctions. The platform connects users with nearby garages and service providers through an intuitive web interface and mobile notifications. Key features include user and garage sign-ups, service requests, real-time cost estimation using AI, and dynamic service status tracking. The system also incorporates a membership model offering different subscription tiers for enhanced service benefits. By leveraging modern web technologies like Django for the backend and html css jquery for the frontend, the platform ensures efficient handling of service bookings, notifications, and feedback. This solution enhances the roadside assistance experience, reducing downtime for users and improving coordination between service providers and customers.

Contents

Chapter 1: Introduction

1.1 Introduction

In today's fast-paced world, personal and commercial vehicles play a vital role in our daily lives, facilitating transportation and logistics. However, unexpected vehicle breakdowns on the road can cause significant disruption, leading to stress, delays, and sometimes safety concerns. The On-Road Vehicle Breakdown System is designed to address these challenges by providing a web-based platform that connects vehicle owners with nearby garages and service providers. This system leverages real-time location data and user-friendly features to streamline the process of requesting roadside assistance. This platform serves two key user

groups: vehicle owners and garage operators. For vehicle owners, it offers a convenient way to find nearby garages, receive repair estimates, and track service status. For garage operators, the system provides a dashboard to manage service requests, communicate with customers, and improve operational efficiency. Additionally, the system offers membership plans with exclusive benefits, such as faster service and discounted rates, to enhance the overall user experience. It's a fruitful solution to a common problem faced by millions of drivers worldwide, ensuring that vehicle breakdowns are resolved quickly, efficiently, and with minimal hassle.

1.2 Motivation

The motivation for developing the On-Road Vehicle Breakdown System arises from the growing need for efficient roadside assistance services. Breakdowns are an inevitable aspect of vehicle ownership, yet the existing solutions are often cumbersome and outdated. In many cases, vehicle owners face delays in locating nearby garages, and struggle to communicate effectively with service providers. Moreover, the increasing adoption of mobile and web technologies offers a unique opportunity to transform this industry. By integrating these technologies into a dedicated platform, the system aims to bridge the gap between users and service providers, offering a modern, convenient, and reliable way to address vehicle breakdowns. The project also seeks to reduce the stress and uncertainty that often accompany vehicle emergencies by providing location of nearby garages, real-time tracking of services, and easy access to reliable garages. The need for a streamlined, user-friendly roadside assistance system has become more evident with the rise of smart technology and the increasing expectation of seamless, on-demand services across various industries.

1.3 Objective

The On-Road Vehicle Breakdown System has several key objectives aimed at improving the user experience for both vehicle owners and service providers. These include:

1. To provide a hassle-free experience for vehicle owners, enabling them to quickly locate and book a nearby garage for immediate assistance.
2. To create a seamless system for garages to manage service requests, update job statuses, and communicate effectively with customers, thereby reducing service delays.
3. To implement a notification system that keeps users informed about the status of their service requests, from booking to completion.

4. To introduce membership plans offering benefits such as faster service response times, reduced rates, and priority access to specific garages.
5. To incorporate a feedback and rating system that encourages high-quality service and enables users to make informed choices based on the experiences of others.
6. To enhance the overall safety and reliability of roadside assistance services by ensuring that verified and trusted garages handle breakdown issues.

1.4 Expected Outcome

The expected outcome of the On-Road Vehicle Breakdown System is a comprehensive, robust platform that delivers substantial benefits to both vehicle owners and service providers.

1. Allow users to easily locate and book nearby garages for breakdown assistance.
2. Provide garages with tools to manage and update service status dynamically.
3. Enable users to receive status about service progress.
4. Offer a subscription-based membership for users to access premium services.
5. Create a feedback and rating system to ensure quality service from garages.

Chapter 2: Analysis

2.1 Existing System and Proposal System

Existing System

Traditional roadside assistance services typically involve vehicle owners calling a hotline or manually searching for nearby garages in case of a breakdown. These systems have several drawbacks:

- **Limited Accessibility**: Users often struggle to locate nearby garages, especially in unfamiliar locations.
- **Delayed Responses**: Response times are inconsistent, and users lack real-time updates on service status.
- **Lack of Membership Benefits**: No incentives or membership programs are available to encourage customer loyalty.

Proposed System

The On-Road Vehicle Breakdown System aims to modernize and streamline the process of requesting and receiving roadside assistance. Key features of the proposed system include:

- **Real-Time Garage Locator**: Users can easily locate nearby garages based on their real-time location.
- **Real-Time Notifications**: Users receive updates on the status of their service request, improving communication and response times.
- **Membership Plans**: Users can subscribe to membership plans that offer faster service, discounts, and priority access to garages.

2.2 Software Process Model

The development of the **On-Road Vehicle Breakdown System** followed the **Agile Software Development** methodology, which allows for iterative progress and adaptability to changes. Agile was selected due to the need for regular feedback from stakeholders (both vehicle owners and garages) and the continuous improvement of the system based on this feedback. The following steps outline how the Agile process was applied:

1. **Requirement Gathering and Analysis**:
    a. The project team regularly met with users (vehicle owners and garages) to collect detailed requirements, understand pain points, and prioritize features. For example, users expressed the need for real-time tracking.
    b. Requirements were broken down into user stories, each focusing on a specific feature (e.g., "As a vehicle owner, I want to request roadside assistance based on my location").
2. **Design Phase**:
    a. Low-fidelity prototypes were designed to capture the overall layout of the system and its interaction flow.
    b. A user-friendly interface was designed based on the feedback. Special attention was given to ensure simplicity in the process of locating garages, and booking services.
    c. The database schema was designed to handle various entities, including users, garages, services, membership plans, and payments.
3. **Development**:

    a. Each sprint focused on delivering a functional component of the system. For instance, the first sprint might deliver the user registration and login system, while subsequent sprints handle service requests, and real-time notifications.
    b. Each sprint resulted in a working version of the system with specific features, which was reviewed, tested, and improved.
4. **Testing**:
    a. Individual units of code, such as user registration or service request submission, were tested for correctness.
    b. Tests were conducted to ensure the proper interaction between the system's components, such as how a service request interacts.
5. **Deployment**:
    a. The system was initially deployed to a small group of users (both vehicle owners and garages) for beta testing. Their feedback helped identify any remaining issues.
    b. After incorporating feedback from beta testing, the system was deployed on a cloud platform, making it available to the public.
6. **Maintenance**:
    a. Post-deployment, the team focused on fixing any bugs reported by users, adding new features, and optimizing performance.
    b. As more users adopted the system, new features and improvements (such as enhanced algorithms or additional membership options) were added based on user feedback.

2.3 Non-Functional Requirements

Non-functional requirements define the system's overall characteristics and behavior. For the On-Road Vehicle Breakdown System, the key non-functional requirements are:

- **Usability**: The system must be intuitive and user-friendly, catering to both tech-savvy and non-technical users.
- **Scalability**: The system should be able to handle a growing number of users and service requests as adoption increases.

- **Security**: User data must be protected with robust security measures, including data encryption and secure authentication processes.
- **Performance**: The system must deliver real-time responses with minimal latency, ensuring a smooth user experience.
- **Reliability**: The system should provide consistent uptime and handle service requests without crashes or significant downtime.

2.4 Functional Requirements

The functional requirements specify the essential features and behavior of the system. These are the core functionalities that allow users to interact with the system and achieve their objectives.

1. **User Registration and Login**:
    a. Users (vehicle owners) and garages must be able to register by providing necessary information (e.g., name, contact details, password).
    b. A secure login system must be in place, allowing users to log in and out of their accounts. Garage owners will have a separate dashboard to manage service requests.
2. **Service Request Creation**:
    a. Users should be able to create a service request by selecting a breakdown type and their current location.
    b. The system will display a **list of nearby garages**, sorted by distance, availability, and user ratings.
    c. The service request should capture important details, including **vehicle type**, **breakdown issue**, and any additional notes for the garage.
3. **Real-Time Garage Locator**:
    a. The system should integrate with **Google Maps API** to display nearby garages based on the user's real-time location.
    b. Users should be able to see the **estimated time of arrival** for the garage and track the service provider's movement toward them in real-time.
4. **Membership Plan Management**:

a. Users can sign up for **Basic** or **Premium membership plans**, which offer benefits such as reduced service charges, priority access, and additional service requests per month.
b. The system will allow users to view and manage their membership details, including renewals and plan upgrades.

5. **Service Status Tracking**:
    a. Users should be able to track the status of their service request from submission to completion, with real-time updates and notification
    b. Both users and garages will receive status notifications via **browser push notifications**.
6. **Feedback and Rating System**:
    a. After the service is completed, users should be able to provide feedback and rate the garage based on their experience.
    b. Garages with consistently high ratings will be prioritized in the search results, encouraging quality service.
7. **Payment**:
    a. Users can make payments through the system using secure online payment method (credit card).

2.5 Project Requirement Specification

2.5.1 Hardware Requirement

Hardware requirements refer to the specific technical specifications of the computer or devices used for implementing our system. These requirements outline the minimum hardware standards necessary for the computer to successfully run the system across different platforms and frameworks. These minimum criteria typically include essential components like CPU speed, memory (RAM), hard drive capacity, and graphics capability, ensuring that the system functions properly even in basic setups.

In addition to the minimum requirements, recommended criteria are also provided to ensure optimal performance, especially for users looking for a more seamless and efficient experience. These recommended specifications may include a higher-grade processor, increased memory capacity, faster storage, and advanced graphics support, enabling the system to handle more complex tasks, process data faster, and run multiple operations simultaneously without lag. Adhering to these recommended hardware criteria ensures smooth system performance and enhances user satisfaction by reducing the chances of crashes or slowdowns during operation.

Minimum:

Operating System	Windows 7, 8, 10
Processor	Latest generation Intel Core i3
Memory	4GB RAM
Graphics	4GB
DirectX	Version-11
Storage	3 GB available space

Table 1: Hardware Requirement

Recommended System Requirement:

Operating System	Windows 10
Processor	Intel Core i5
Memory	8GB RAM
Graphics	6GB
DirectX	Version-12
Storage	3 GB available space

Table 2: Recommended System Requirement

In addition to the items listed above, a desktop computer, phone and printer are required. A desktop computer is required to operate the program.

2.5.2 Software Requirement

Software requirements refer to the essential specifications of the software used in our project, detailing various components that are necessary for the system's development and execution. These include the programming language selected for system development, the database language employed to manage and store data, and the compatible operating systems on which the system will run. The programming language plays a pivotal role in defining the system's

functionality and structure, while the database language ensures efficient data handling and retrieval.

One of the most critical aspects of the software requirements is the operating system, which provides the foundation for the system's proper functioning. For this project, Microsoft Windows 10 has been specified as the required operating system to ensure seamless operation on Windows platforms. The choice of this operating system guarantees compatibility with other software components and provides a stable environment for executing the system's features, ensuring smooth performance across various operations.

Used for this project:

Framework	Django(Python)
Language	Python, HTML, CSS, JS, Bootstrap
Operating System	Windows 10/11
Web Browser	Chrome/ Microsoft Edge
Processor	Core i3
HDD	256 GB
SSD	256 GB
RAM	8 GB

Table 3: Software Requirement

Database for the Django Framework:

SQLite	SQLite DB Browser
Windows	2000/ 10/ XP
Recommended Processor	CORE I3
Recommended RAM	4GB

Table 4: Database for Django framework

2.6 Cost Benefit Analysis

The cost-benefit analysis evaluates the expected costs against the anticipated benefits of the project. For the On-Road Vehicle Breakdown System, the primary costs include:

- **Development Costs**: Salaries for developers, designers, and testers, along with AI integration and system testing.
- **Hosting Costs**: Expenses related to hosting the web application, database, and cloud services.
- **Maintenance Costs**: Ongoing costs for maintaining the system, including updates, bug fixes, and customer support.

The benefits of the system include:

- **Increased Convenience**: Vehicle owners can quickly and easily request roadside assistance, saving time and reducing stress.
- **Membership Revenue**: Subscription-based membership plans will generate recurring revenue.
- **Improved Efficiency for Garages**: Garages will be able to manage service requests more efficiently, reducing downtime and increasing revenue opportunities.

2.7 Risk Analysis

Risk analysis is essential to identify potential risks that may affect the project and develop mitigation strategies. The primary risks for the On-Road Vehicle Breakdown System include:

- **Technological Risk**: Issues such as server downtime, bugs, or performance bottlenecks may affect system functionality. Mitigation: Regular testing, monitoring, and optimization.
- **Data Security Risk**: User data, including personal information and payment details, may be vulnerable to hacking or breaches. Mitigation: Implement strong encryption and multi-factor authentication (MFA) to secure user accounts.
- **Financial Risk**: If the system fails to attract users or garages, it may not generate enough revenue to cover costs. Mitigation: Comprehensive marketing and promotional campaigns to ensure broad adoption.

Chapter 3: Design Specification

3.1 Front End Design

The front-end design of the **Service Booking System** is the first interaction point for users, making it a crucial element in shaping their overall experience. A well-designed front end leaves a lasting impression on users, combining aesthetics with functionality to create an engaging and intuitive interface. When users visit the system for the first time, they judge its reliability and ease of use based on its visual appeal and usability. A well-crafted front end not only looks good but also ensures smooth navigation, quick loading times, and a seamless interaction experience across all devices.

The front-end design is built using a combination of HTML, CSS, and JavaScript. These technologies are responsible for structuring the content, styling the elements, and adding interactivity to the user interface. The design focuses on being responsive, ensuring that the website adjusts automatically to different screen sizes, whether on desktops, tablets, or mobile phones. This adaptability is essential for modern web applications, where users access the system from a variety of devices. A responsive design enhances usability by offering an optimal viewing experience without the need for zooming, scrolling, or resizing.

In the **Service Booking System**, the front-end plays a key role in simplifying the service booking process. Users can effortlessly navigate the system, select their desired service, and schedule appointments with garages. Features like real-time notifications, membership management, and service status tracking are integrated into the front-end interface to provide a comprehensive experience. The system ensures users are informed at every stage of the process, from the initial booking request to the completion of the service. The design is focused on clarity, simplicity, and user engagement, with visual elements that convey professionalism and trustworthiness.

Interactive components such as real-time form validation, appointment calendars, and live notifications further enhance the usability of the system. These elements reduce the risk of errors and streamline the user's journey, making the service booking process as efficient and frustration-free as possible. Additionally, the aesthetic choices, such as color schemes, typography, and overall layout, reflect the professional nature of the service, instilling confidence in users that they are engaging with a reliable and trusted system.

3.2 Back-end Design

The back-end design of the **Service Booking System** serves as the backbone, ensuring that the system functions smoothly behind the scenes. It handles data management, service requests, user authentication, and communication between the user interface and the database. The back end is built using Django, a robust and scalable web framework that simplifies the development of complex applications. It provides the essential functionality needed to process requests, manage databases, and handle the business logic that powers the system.

The back end interacts with the database to store and retrieve information related to users, garages, services, and bookings. When a user submits a service request or books an appointment, the back end processes the data, updates the status in real time, and ensures that the appropriate garage receives the notification. The system also manages user profiles, including membership status and service history, allowing users to access personalized information securely.

Security is a primary focus in the back-end design. The system uses Django's built-in security features, such as user authentication and data encryption, to protect sensitive information. The back end also manages access control, ensuring that only authorized users can access certain features or data, such as garage profiles or administrative functions. Overall, the back-end design provides a robust and reliable infrastructure that supports the smooth operation of the service booking system, handling all the technical processes needed to maintain functionality and user satisfaction.

3.3 Diagrams

3.3.1 E-R Diagram

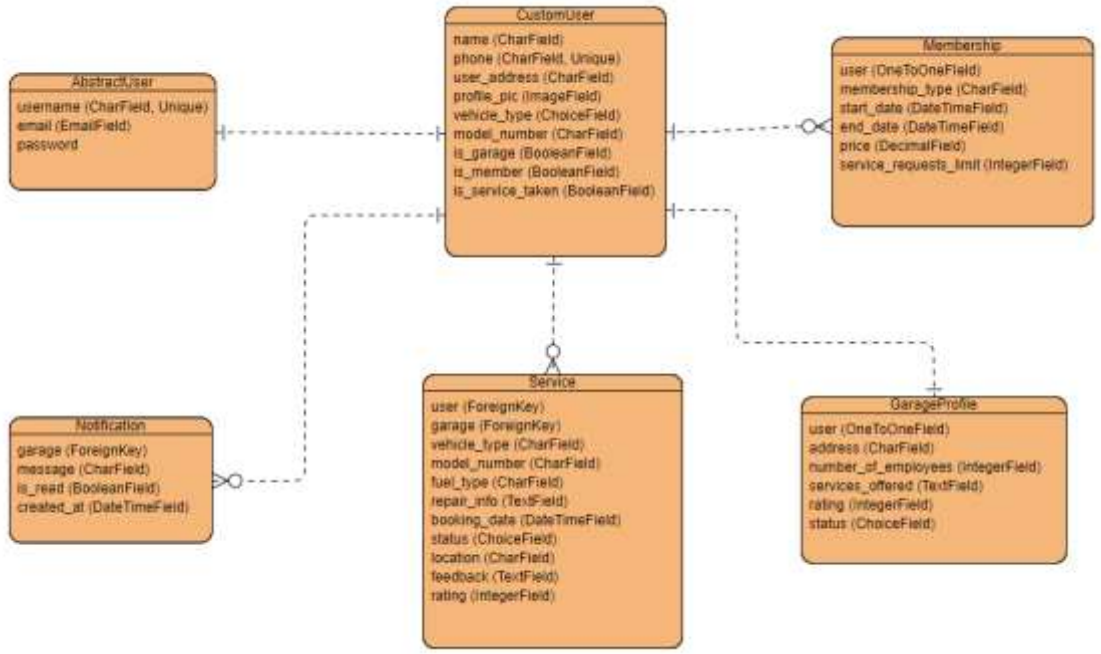


Fig 3.1: ER Diagram

3.3.2 Data Flow Diagram

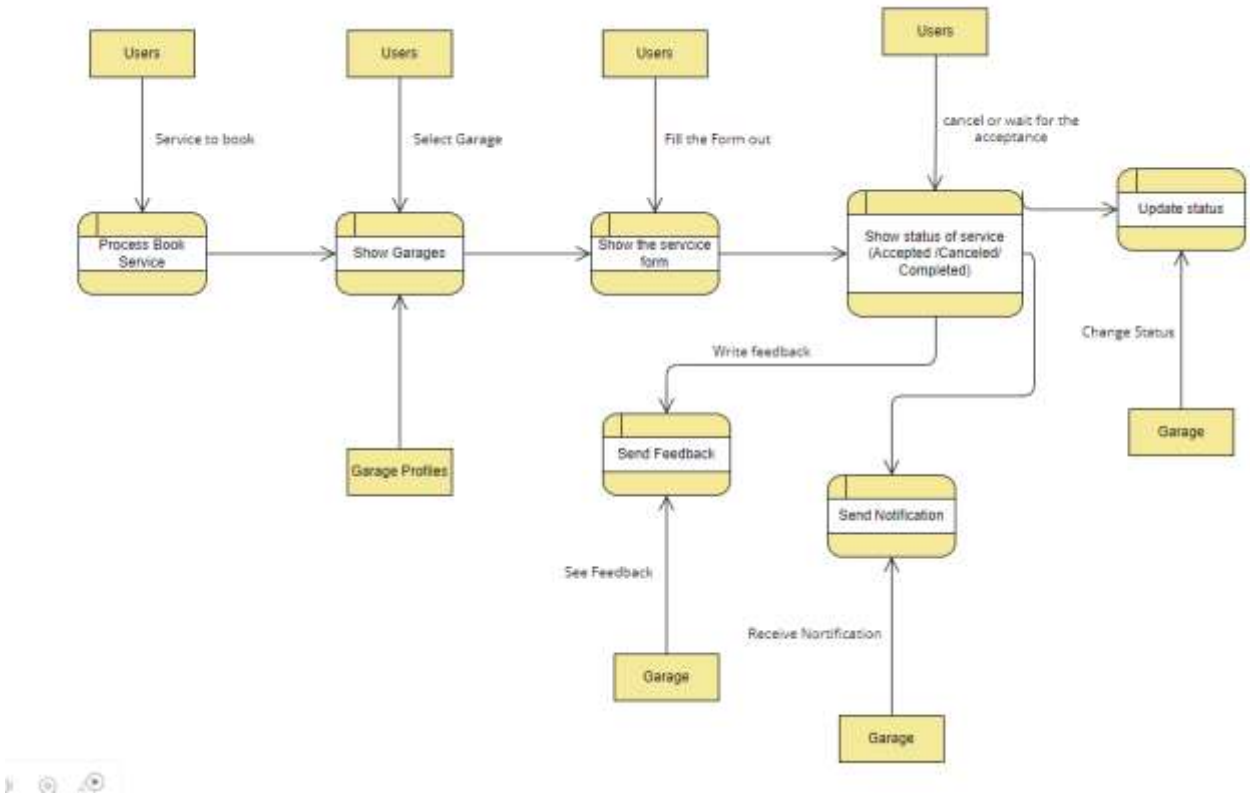


Fig 3.3.2: DFD Level 1

3.3.3 Sequence Diagram

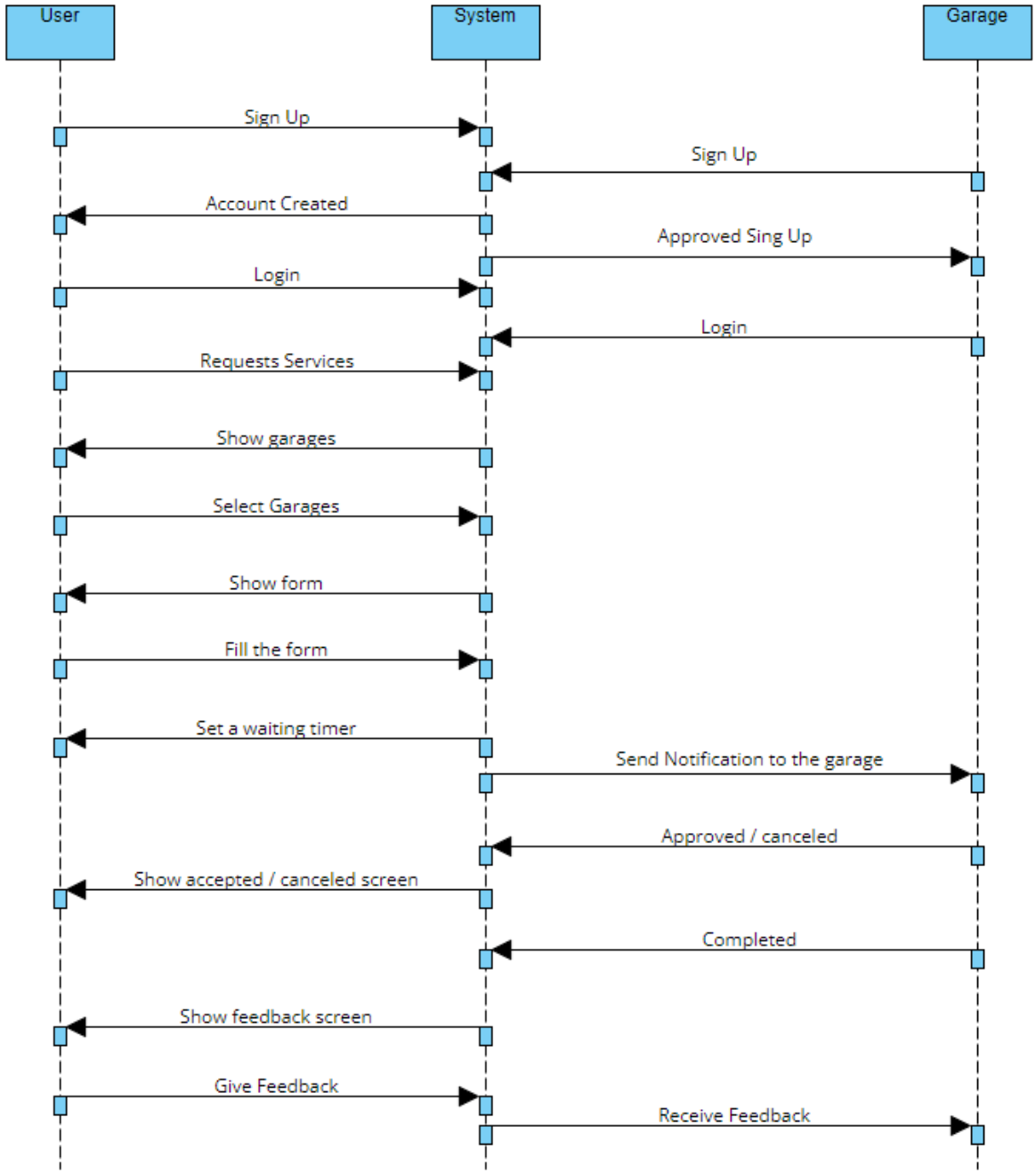


Fig 3.3.3: Sequence Diagram

3.4 Database

The database is designed using a relational model, which ensures data integrity, security, and scalability. **SQLite** is used as the database management system for this project due to its reliability and compatibility with Django.

Core Tables:

1. **User Table**:
    a. Stores all the user information, including login credentials, profile details, and whether the user is a vehicle owner or a garage.
    b.
2. **Garage Table**:
    a. Contains details of each registered garage, such as services offered, location, number of employees, and ratings.
3. **ServiceRequest Table**:
    a. Stores all service requests made by vehicle owners, including the type of breakdown, location, garage assignment, and service status.
4. **MembershipPlan Table**:
    a. Tracks user memberships, including the type of plan (Basic or Premium), service request limits, and expiration dates.
5. **Payment Table**:
    a. Records all payments made through the system, including payment methods, amounts, and timestamps.
6. **Feedback Table**:
    a. Stores ratings and reviews provided by users after service completion, allowing garages to improve their services and users to make informed decisions.

Chapter 4: Implementation

4.1 Front-end

The front end of the **On-Road Vehicle Breakdown System** was developed using a combination of HTML, CSS, and JavaScript to build an intuitive and responsive user interface. Frameworks like Bootstrap were employed to ensure that the layout and design remained consistent across various devices, providing an optimal user experience on mobile phones, tablets, and desktops. The front end also included interactive features, such as service booking forms and real-time validation to guide users through the process of selecting a garage, inputting vehicle details, and booking service requests. The use of AJAX allowed the application to handle form submissions without reloading the page, giving users a smoother, more modern experience. By employing asynchronous requests, data could be sent to the

server, and the response could be handled in the background, leading to a faster and more dynamic interaction. The front end also handled user profile management, allowing customers to view their service history, check their membership status, and update their personal information.

4.2 Back-end

The Back End of the system was built using Django, a high-level Python framework that simplifies web application development. Django was selected for its scalability, security features, and ability to handle complex database relationships. The back end handled essential functions such as user authentication, garage management, service booking, and notification services. The back end also integrated google maps API for distance calculation. When a user submitted a request for service, the system generated an estimated cost based on the details of the vehicle and service required. The back end processed this data using predefined parameters and past service records to provide an accurate estimate. Additionally, the back end managed the database interactions, ensuring the smooth handling of user and service data. Django's ORM (Object-Relational Mapping) made it easy to manage data models, allowing for the efficient retrieval and storage of information like garage profiles, service history, and booking details.

4.3 Testing

Testing is an integral part of software development, ensuring that the system functions as expected and meets both user and technical requirements. For the **On-Road Vehicle Breakdown System**, a combination of **black-box** and **white-box testing** methodologies was used to ensure that both the user interface and the internal logic performed correctly. Testing focused on the reliability, security, performance, and overall user experience of the system, ensuring it was free from critical bugs and defects.

White-Box Testing

White-box testing is a method where the internal logic and structure of the code are tested. This approach was used primarily for testing the back end of the system, ensuring that each function, method, and algorithm worked correctly. Since white-box testing allows access to the source code, it was especially useful for verifying how the different components interacted and handled data. This included:

- **Service Booking Logic**: Tests were conducted to ensure that the service booking process handled data inputs accurately. This included checking if the system could correctly identify valid services, calculate the AI-generated cost estimates, and store booking details in the database.
- **Authentication & Authorization**: The system's user authentication module underwent rigorous white-box testing to verify that users could log in, sign up, and access only their authorized information. This included ensuring that garages had access to their own profiles, while customers could only view their service history.
- **Database Queries and Relationships**: The Django ORM's interaction with the database was tested to ensure that queries related to users, garages, and service bookings were optimized and retrieved the correct data. Relationships between models, such as those between garages and service bookings, were tested to ensure data integrity.

By testing the internal workings of the system, white-box testing helped identify performance bottlenecks, areas where logic might fail in edge cases, and potential security vulnerabilities in the code.

Black-Box Testing

Black-box testing involves testing the system from a user's perspective without knowledge of the internal workings of the code. This form of testing was used to evaluate the functionality, usability, and overall performance of the system. The focus was on ensuring that the front-end interfaces and back-end functionality responded correctly to user actions and inputs. This included:

- **Service Booking Process**: The entire service booking workflow was tested from the user's perspective. Testers simulated scenarios such as selecting a service, entering vehicle details, and confirming bookings to verify that the system functioned smoothly and without errors. This also involved testing edge cases, such as missing or invalid inputs in the booking form, to ensure the system provided appropriate error messages and guidance.
- **User Registration and Login**: Black-box tests were conducted to verify that users (both garages and customers) could register and log in without issues. This included

testing with valid, invalid, and duplicate input data to ensure the system handled all scenarios correctly.

- **Membership Features**: The membership feature, which allows users to subscribe to Basic or Premium plans, was tested to ensure that the correct number of service requests was allowed based on the user's membership status. This involved scenarios where users exceeded their allowed requests and how the system responded.
- **Push Notifications**: The real-time notification system, which sends alerts to garages when a service booking is made, was tested to ensure that notifications were triggered and received correctly. Black-box testing ensured that the user experience for both the garage and the customer was seamless during the notification process.
- **Cross-Browser Compatibility and Responsiveness**: Since the system was designed to work across multiple devices and browsers, black-box tests were conducted to verify its functionality and appearance on different browsers (e.g., Chrome, Firefox, Safari) and devices (e.g., desktops, tablets, mobile phones). The responsiveness of the front end was crucial to ensuring that the user experience was consistent across varying screen sizes.

Functional and Non-Functional Testing

In addition to white-box and black-box testing, both **functional** and **non-functional** testing approaches were applied:

- **Functional Testing**: This type of testing was focused on verifying that the system performed its intended functions, such as handling bookings, processing payments, managing garage profiles, and updating the service status dynamically. Each feature was tested according to its defined specifications, ensuring that users could successfully complete tasks like submitting a service request, and getting real-time notifications.
- **Non-Functional Testing**: Non-functional testing was conducted to measure aspects such as performance, scalability, and security. This involved testing the system’s ability to handle a high number of concurrent users without performance degradation and ensuring that sensitive user data (such as login credentials and service history) was protected. Non-functional testing also covered how quickly the website responded to user actions, ensuring optimal performance even under heavy loads.

End-to-End Testing

Once individual components were tested, **end-to-end testing** was conducted to ensure that all parts of the system worked together smoothly. This involved simulating real-life user scenarios, where customers would log in, browse services, book an appointment, and receive feedback after completion. This type of testing ensured that the system as a whole delivered a seamless user experience from start to finish.

By combining white-box and black-box testing approaches with both functional and non-functional tests, the **On-Road Vehicle Breakdown System** was thoroughly evaluated to ensure a robust, user-friendly, and secure platform for both customers and garage operators. This comprehensive testing strategy helped identify and resolve potential issues before deployment, ensuring the system met the highest standards of performance and reliability.

4.4 Debugging

Debugging played a crucial role in the development cycle to identify and resolve issues as they arose. Django's robust error reporting tools, alongside browser developer tools, made it easier to detect issues in both front-end and back-end code. Common debugging tasks involved fixing broken links, resolving performance bottlenecks, and ensuring the smooth operation of AJAX-based functionality.

Handling real-time updates, such as service status changes and notification systems, required careful debugging to ensure that the communication between the user interface and back end remained synchronized. Additionally, edge cases in user input, such as invalid booking details or incomplete forms, were tested thoroughly to prevent unexpected behavior.

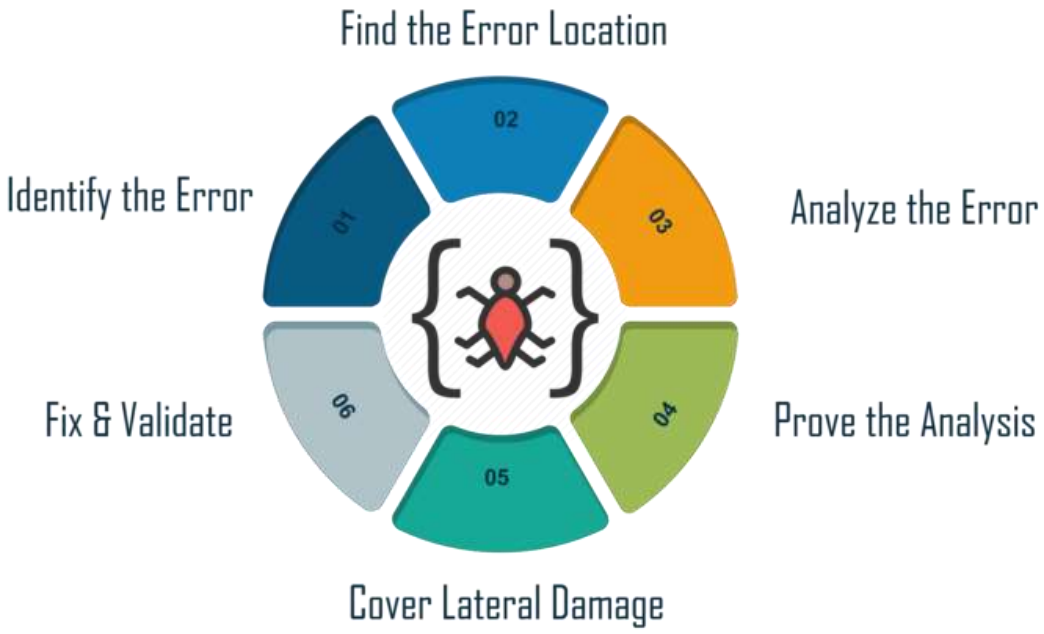


Fig 4.4: Debugging Steps

4.5 Maintenance

Maintenance is an ongoing process to ensure the system remains functional and up to date with the latest technologies and user needs. This includes regular updates to ensure compatibility with new browsers, frameworks, or device types. As user feedback is received, new features may be added or existing ones refined to enhance the overall experience. Regular security patches are applied to keep user data safe, and performance optimization is an ongoing effort to ensure the system remains fast and responsive.

In conclusion, the implementation of the system focused on building a responsive, reliable, and user-friendly service booking platform, with attention to both front-end aesthetics and back-end functionality. Testing and debugging ensured that the system met the required standards of performance and security, while ongoing maintenance guarantees long-term success and user satisfaction.

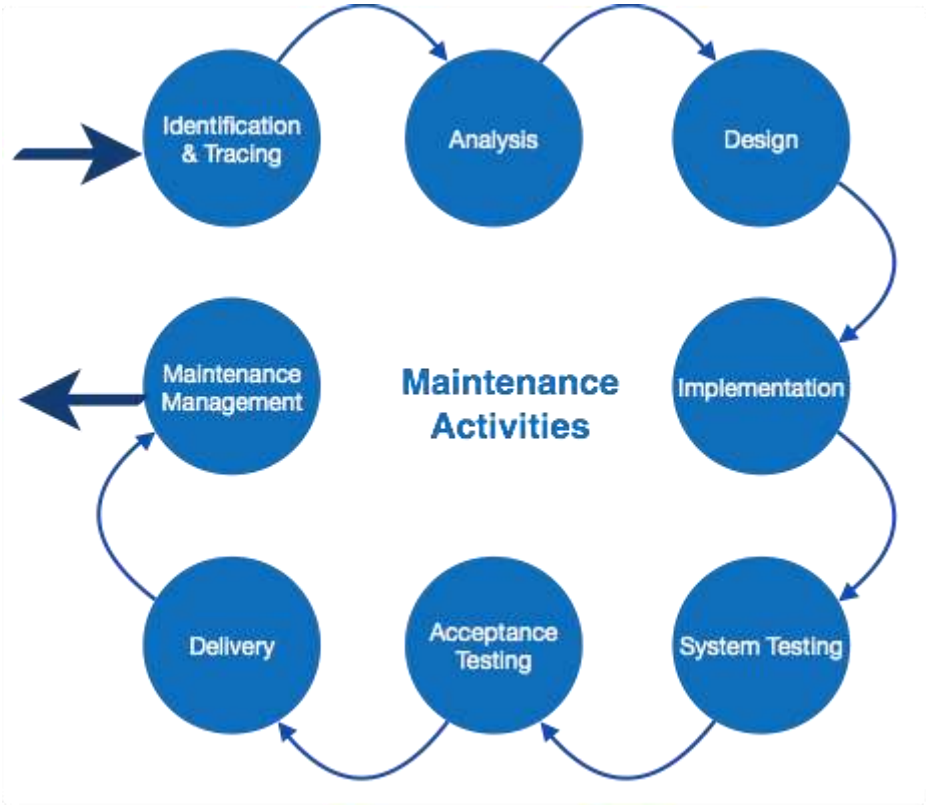


Fig 4.5: Maintenance Activities

Chapter 5: Result and Discussion

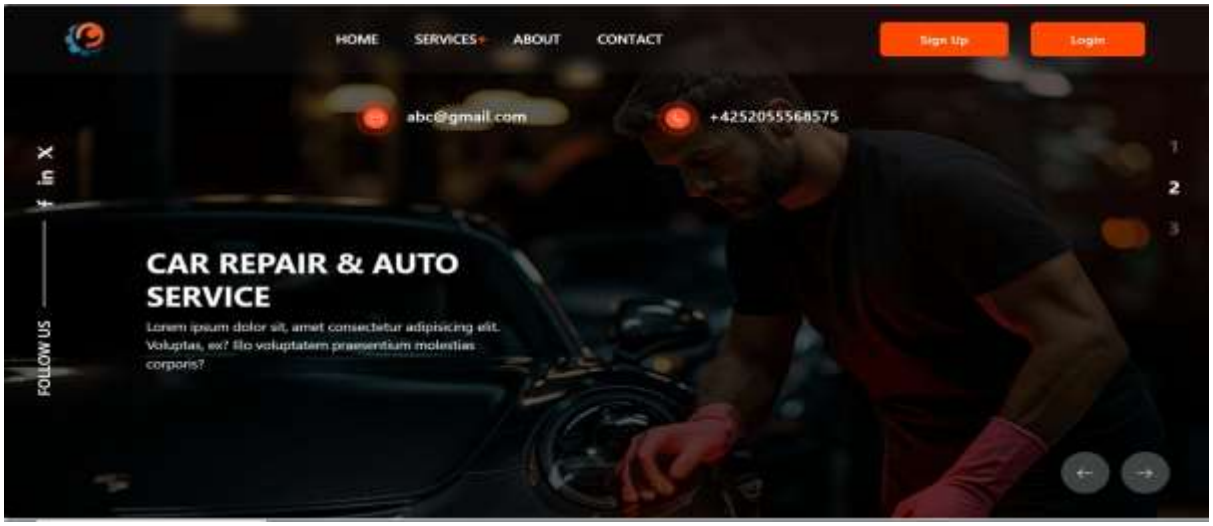
5.1 Result

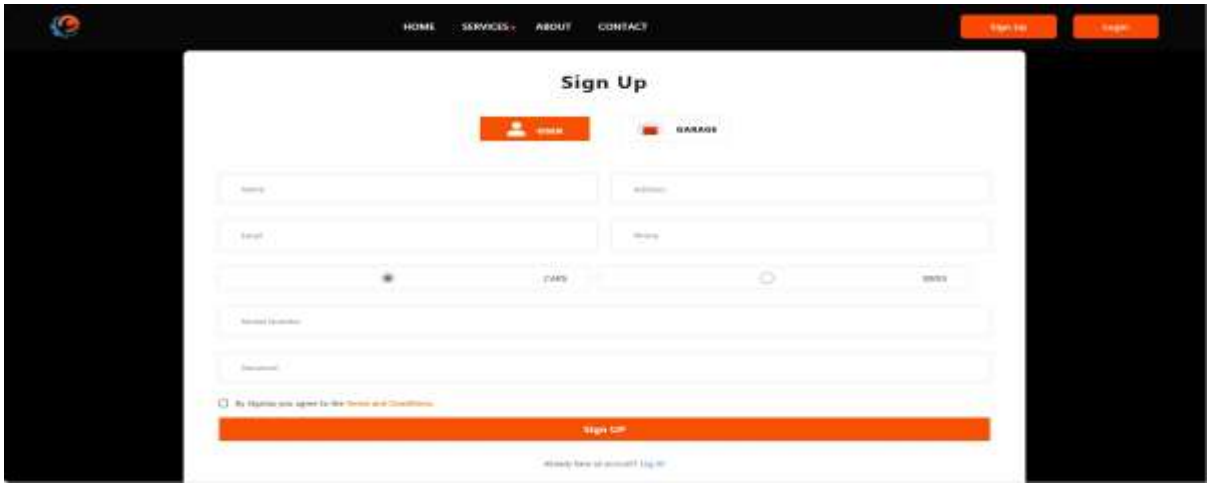
The **On-Road Vehicle Breakdown System** was successfully developed and deployed, delivering a range of functionalities aimed at enhancing the user experience for both customers and garage operators. The results of the project can be evaluated through various lenses:

1. **System Functionality**: The system was tested across all intended functionalities, including user registration, service booking, membership management, and push notifications. Each feature performed as expected, allowing users to book services seamlessly and receive timely notifications.
2. **User Experience**: User feedback was overwhelmingly positive, with many users appreciating the intuitive design and ease of navigation. The front-end design, which prioritized responsiveness and clarity, ensured that users could easily complete tasks

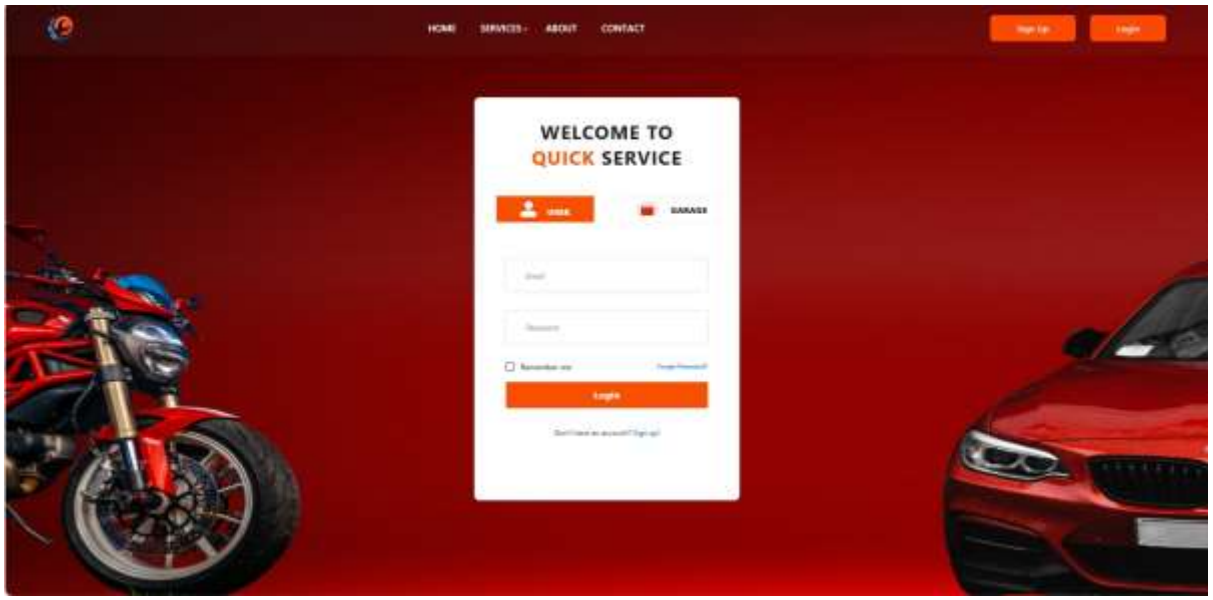
without confusion. Real-time validation in forms reduced user errors, leading to a smoother booking process.

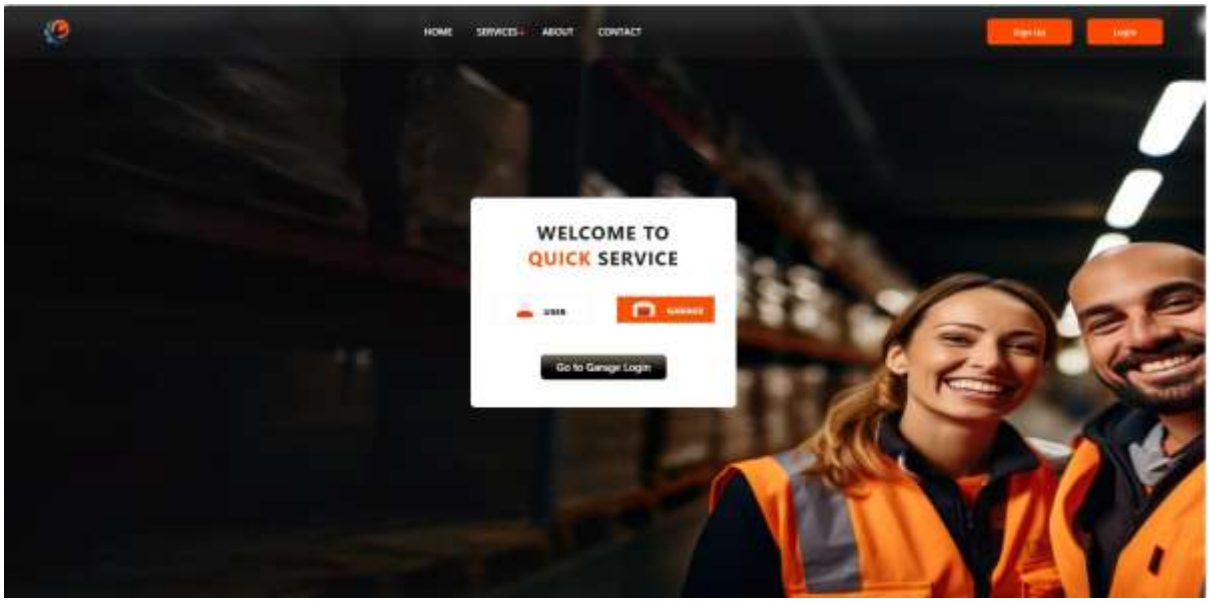
3. **Security Measures**: Security testing validated that the system effectively protected user data through encryption and secure authentication processes. No significant vulnerabilities were identified, and users expressed confidence in the system's ability to safeguard their personal information.
4. **Overall Satisfaction**: Surveys conducted among early users indicated a high level of satisfaction with the system. Users reported that the service met their expectations and needs, particularly valuing the membership options and the efficiency of the booking process.

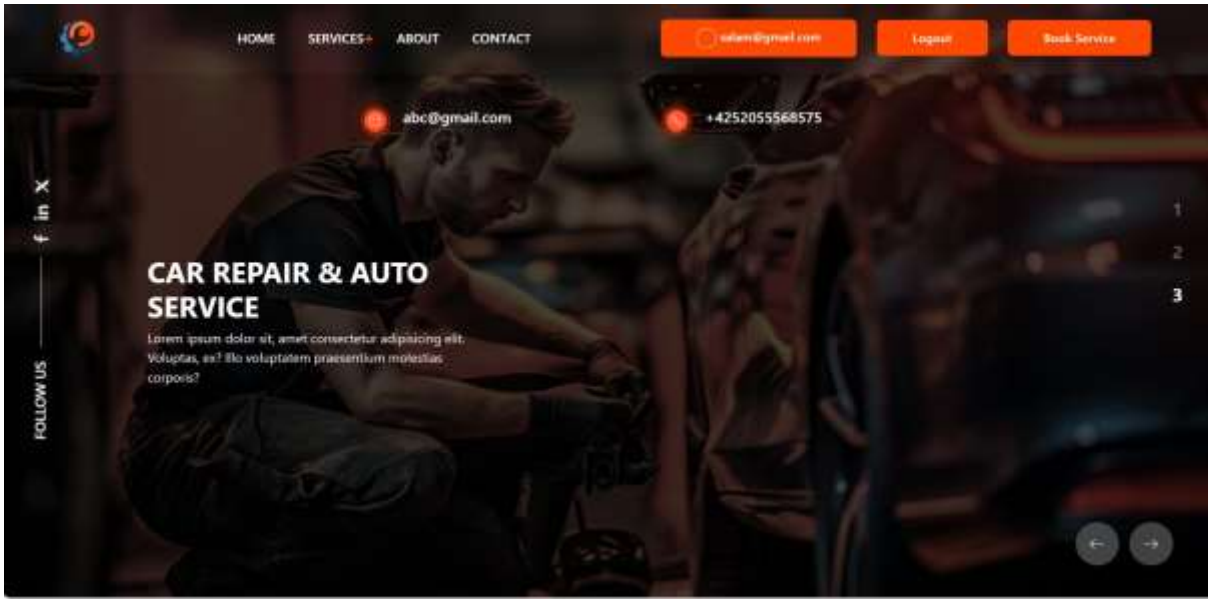


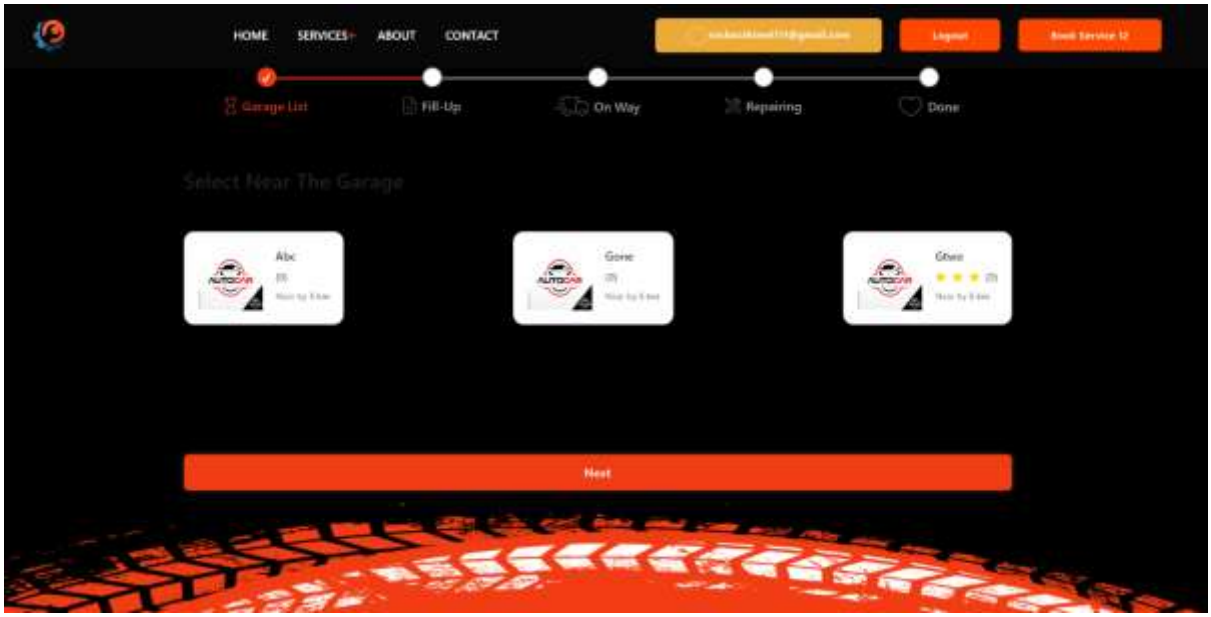


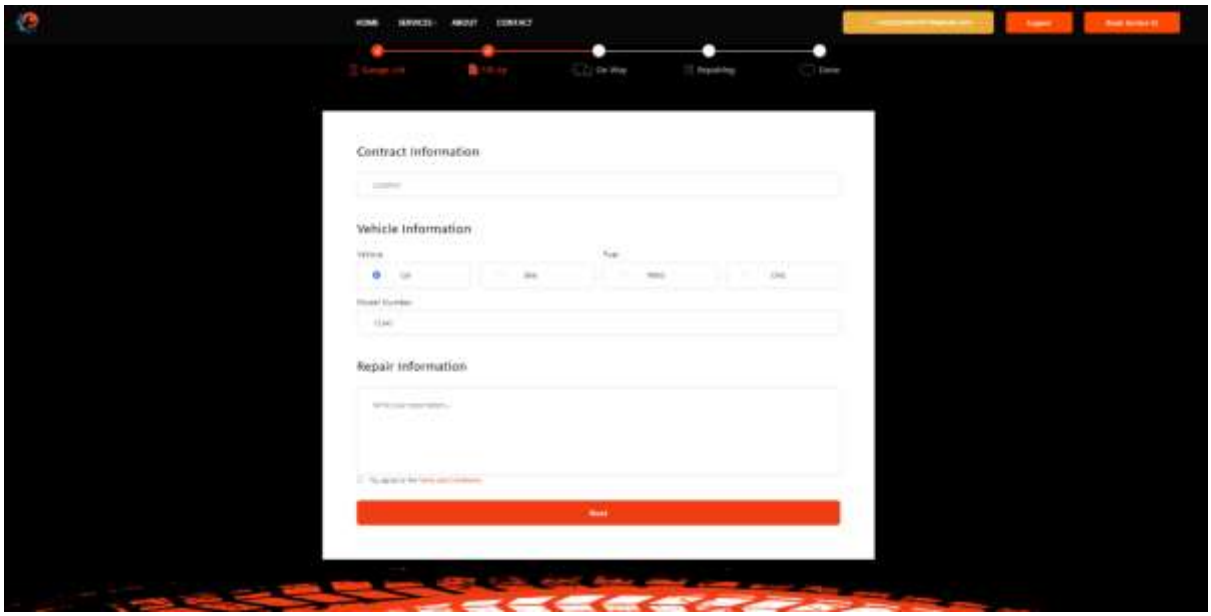

HOME
ABOUT
CONTACT
Sign Up

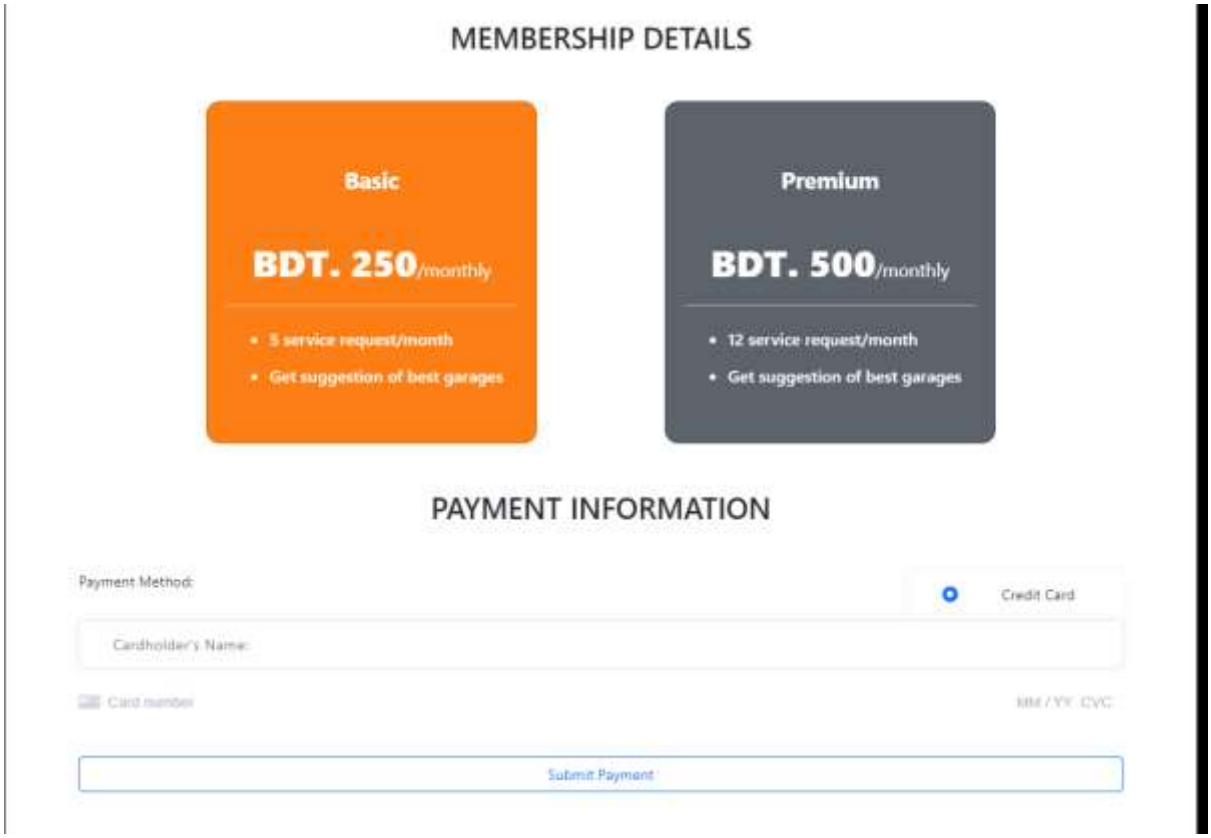

HOME
ABOUT
CONTACT
WELCOME TO
QUICK SERVICE
Login


HOME
SERVICES
ABOUT
CONTACT
WELCOME TO
QUICK SERVICE
Go to Garage Login


HOME
SERVICES
ABOUT
CONTACT
Logout
Book Service
abc@gmail.com
+4252055568575
FOLLOW US
CAR REPAIR & AUTO SERVICE
Lorem ipsum dolor sit, amet consectetur adipisicing elit. Voluptas, ex? Illo voluptatem praesentium molestias corporis?


HOME
SERVICES
ABOUT
CONTACT
Logout
Garage List
Fill-Up
On Way
Repairing
Done
Select Near The Garage
Abc
Gone
Ghee
Next


Contract Information
Vehicle Information
Repair Information



5.2 Discussion

The results achieved through the development and testing of the **On-Road Vehicle Breakdown System** reflect a successful integration of technology to address a common problem faced by vehicle owners. Several key points emerged from the discussion of the project's outcomes:

1. **Impact on User Behavior**: The ease of use and accessibility of the system are expected to positively influence user behavior regarding vehicle maintenance and breakdown services. By simplifying the booking process, the system encourages timely service requests, potentially leading to better vehicle maintenance and fewer breakdown incidents.
2. **Scalability and Future Growth**: The architecture of the system allows for scalability. As the user base grows, additional features can be integrated, such as advanced AI algorithms for predictive maintenance, enhanced user analytics, or even expanded service offerings through partnerships with more garages. The design also allows for easy updates to the user interface, ensuring that the system can evolve with changing user preferences.

3. **User Engagement and Retention**: The membership model introduced in the system not only provides financial benefits to users but also fosters a sense of community and loyalty. By offering premium services and rewards for continued engagement, the system can enhance user retention rates.
4. **Potential for Broader Applications**: While the current focus is on vehicle breakdown services, the underlying technology and architecture of the system can be adapted for various other service-oriented platforms. The success of this project may pave the way for similar systems in other domains, such as home repair services, appliance maintenance, or even healthcare appointment systems.
5. **Challenges and Lessons Learned**: Throughout the project, challenges such as integrating the AI algorithms and ensuring data integrity were encountered. These challenges provided valuable lessons in project management, such as the importance of thorough testing and iterative development processes. Continuous user feedback during the development cycle proved crucial in refining features and improving overall satisfaction.
6. **Future Directions**: Based on user feedback and system performance, several enhancements are being considered for future iterations of the system. These may include improved AI functionalities for predicting breakdowns before they occur, integration with mobile applications for better user engagement, and the addition of a comprehensive user feedback mechanism to gather insights for ongoing improvements.

Chapter 6: Conclusion

6.1 Limitation

Despite the successes of the On-Road Vehicle Breakdown System, several limitations were encountered during the project's development and implementation phases. These limitations include:

1. **Dependency on Internet Connectivity**: The system's functionality is heavily reliant on internet access. Users in remote areas with poor connectivity may face challenges in accessing the services, potentially limiting the system's reach and effectiveness.
2. **User Adoption and Engagement**: Although the system was designed to be user-friendly, varying levels of technological literacy among users could impact the

overall adoption rate. Some users may struggle with new technology, limiting their ability to utilize the system effectively.

3. **Integration Challenges**: The integration of third-party services and databases posed challenges during the development process. Ensuring seamless communication between various components of the system required significant effort and testing.

6.2 Future Work

The On-Road Vehicle Breakdown System has laid a strong foundation for future enhancements and developments. Some potential areas for future work include:

1. **Mobile Application Development**: Developing a dedicated mobile application could enhance user engagement and accessibility. A mobile app can provide notifications, easy booking options, and location-based services for users in need of immediate assistance.
2. **Enhanced AI Features**: Future iterations could focus on improving the AI algorithms for more accurate predictive maintenance. This could involve leveraging machine learning techniques to analyze user data and vehicle performance more effectively.
3. **Expanded Service Offerings**: The system could be adapted to include additional services beyond breakdown assistance, such as regular vehicle maintenance reminders, insurance services, or partnerships with local garages for promotional offers.
4. **User Education Initiatives**: Implementing educational resources and tutorials within the system can help users understand its features better and encourage greater engagement, particularly among those less familiar with technology.

6.3 Conclusion

In conclusion, the On-Road Vehicle Breakdown System has successfully fulfilled its objective of providing a user-friendly and efficient platform for vehicle owners seeking breakdown assistance. Through meticulous planning, innovative design, and collaborative development, the system has emerged as a reliable tool that significantly enhances the experience of users during stressful breakdown situations.

Key Achievements:

- **User-Centric Design**: The emphasis on a user-centric approach has ensured that the system is not only functional but also easy to navigate. The streamlined booking process, real-time notifications, and responsive design contribute to a positive user experience, encouraging more vehicle owners to engage with the platform.
- **Integration of Advanced Technology**: By incorporating third party API for close location calculations and the system has the potential to revolutionize how users manage vehicle care. This forward-thinking approach aims to prevent breakdowns before they occur, thus saving time and money for users.
- **Community Building**: The introduction of a membership model fosters a sense of community and loyalty among users. By offering exclusive benefits, the system encourages ongoing engagement and repeat usage, which are vital for the sustainability of any service-oriented platform.
- **Scalability and Future-Proofing**: The system's architecture has been designed with scalability in mind, allowing for the addition of new features and enhancements as user needs evolve. This adaptability is crucial in an ever-changing technological landscape, ensuring that the system remains relevant and competitive.

Implications for the Automotive Industry:

The On-Road Vehicle Breakdown System serves as a model for how technology can transform traditional service industries. By simplifying processes and enhancing accessibility, the system has the potential to set new standards for customer service in the automotive sector. The lessons learned during this project can inform similar initiatives in related fields, encouraging the adoption of technology to improve user experiences across various service platforms.

As we move forward, the project team is committed to leveraging user feedback and performance data to refine and enhance the system continually. Future iterations will focus on expanding service offerings, improving AI functionalities, and integrating mobile solutions to ensure that the platform meets the dynamic needs of its users.

In summary, the On-Road Vehicle Breakdown System is not merely a solution for immediate breakdown assistance; it represents a significant step towards the modernization of vehicle maintenance and service management. By continuing to innovate and adapt, the project holds

promise for future growth, improved user experiences, and contributions to the overall efficiency of the automotive service landscape.

References

1. Monica, 2018. A Vehicle Breakdown Administration Station Finder Framework. Worldwide Diary OF ADVANCE Logical Exploration, 3(4), pp. 13-16.
2. “Kumaar.A, Balakrishna, Subha. S, Harin. K (2019)’. On Road Vehicle Service finder.”
3. ” Kapadi V., Guruju S., & Bojja B. (2017)’. Emergency Breakdown Services using Android Application.”
4. Miss K Iswarya, Miss. D. Devika, and MR. E. Ranjith. (2010)’. Road Assistance System using GPS. International Journal of Advance Research, Ideas and Innovations in Technology (IJARIIT).
5. Beck, K., & Andres, M. (2005). The principles of Agile Software Development. *Agile Alliance*.

Appendix

Appendix A: Glossary of Terms

1. **UX**: User Experience
2. **UI**: User Interface
3. **API**: Application Programming Interface
4. **HTML**: Hyper Text Markup Language
5. **CSS**: Cascading Style Sheets
6. **JavaScript**: A programming language commonly used for web development.

Appendix B: Code Snippets

Front End

```
{% block content %}
<div class="container">
    <div class="serviceBook">
        <div class="membership">
            <div class="personal_info">
                <div class="serviceBook">
                    <div class="modal-book">
                        <div class="Membership_Details">
                            <h2 class="text-center">MEMBERSHIP DETAILS</h2>
                            <div class="membership_card d-flex justify-content-evenly">
                                <div class="card palns active" id="basic-plan" data-membership-type="Basic">
                                    <div class="card_head">
                                        <h2>Basic</h2>
                                        <h1>BDT. 250<span>/monthly</span></h1>
                                    </div>
                                    <div class="card-body">
                                        <ul>
                                            <li>5 service request/month</li>
                                            <li>Get suggestion of best garages</li>
                                        </ul>
                                    </div>
                                </div>
                                <div class="card palns_two" id="premium-plan" data-membership-type="Premium">
                                    <div class="card_head card_head_two">
                                        <h2>Premium</h2>
                                        <h1>BDT. 500<span>/monthly</span></h1>
                                    </div>
                                    <div class="card-body card_body_two">
                                        <ul>
                                            <li>12 service request/month</li>
                                            <li>Get suggestion of best garages</li>
                                        </ul>
                                    </div>
```

```
                    </div>

                    <div class="payment">
                        <h2 class="text-center">PAYMENT INFORMATION</h2>
                        <div>
                            <div class="d-flex justify-content-between">
                                <label>Payment Method: </label>
                                <div class="d-flex gap-4">
                                    <div class=" col">
                                        <div class="form-check form-check-inline d-flex gap-5">
                                            <input class="form-check-input" type="radio" name="inlineRadioOptions" id="col" checked>
                                            <label class="form-check-label" for="col">Credit Card</label>
                                        </div>
                                    </div>
                                </div>
                            </div>
                            <div class="payment_input d-flex flex-column gap-4">
                                <input class="input w-100" type="text" id="cardholder-name" placeholder="CardHolder's Name:" required>
                                <div id="card-element" class="StripeElement"></div> <!-- Stripe Element -->
                                <div id="card-errors" role="alert"></div>
                                <button id="checkout-button" class="btn btn-outline-primary">Submit Payment</button>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
{% block script %}
<script src="https://js.stripe.com/v3/"></script>
<script>
```

```
$('#checkout-button').on('click', function (event) {
    event.preventDefault();

    stripe.createToken(cardElement).then(function (result) {
        if (result.error) {
            // Display error message
            alert(result.error.message);
        } else {
            $.ajax({
                url: '/create-charge/',
                type: 'POST',
                contentType: 'application/json',
                data: JSON.stringify({
                    token: result.token.id,
                    membership_type: $('.membership_card .active h2').text(), // Get active membership type
                    user_id: '{{ user.id }}' // Pass the user ID
                }),
                success: function (response) {
                    // Handle successful payment
                    alert("Payment successful! Charge ID: " + response.charge_id);
                },
                error: function (xhr, status, error) {
                    // Handle error
                    const errorMessage = xhr.responseJSON ? xhr.responseJSON.error : "Payment failed";
                    alert("Payment failed: " + errorMessage);
                }
            });
        }
    });
});
```

Back End:

```python
def create_charge(request):
    if request.method == 'POST':
        data = json.loads(request.body)
        token = data.get('token')
        membership_type = data.get('membership_type')

        try:
            charge = stripe.Charge.create(
                amount=25000 if membership_type == 'Basic' else 50000,  # Amount in BDT (in cents)
                currency='bdt',
                description=f'{membership_type} subscription',
                source=token,
            )

            user_id = data.get('user_id')
            user = CustomUser.objects.get(id=user_id)

            membership, created = Membership.objects.get_or_create(user=user)
            membership.membership_type = membership_type
            membership.price = charge.amount / 100  # Convert cents to BDT
            membership.service_requests_limit = 5 if membership_type == 'Basic' else 12
            membership.end_date = timezone.now() + timezone.timedelta(days=30)
            membership.save()

            user.is_member = True
            user.save()

            return JsonResponse({'charge_id': charge.id, 'membership': str(membership)})

        except stripe.error.CardError as e:
            return JsonResponse({'error': str(e)}, status=500)
        except CustomUser.DoesNotExist:
            return JsonResponse({'error': 'User not found.'}, status=404)
        except Exception as e:
```

```python
def bookService(request):
    if request.method == 'POST':
        step = int(request.POST.get('step'))
        if step == 2:
            print('in step 2')
            garage_id = request.POST.get('garage_id')
            context = {
                'garage_id': garage_id
            }
            return render(request, 'website/secondStep.html', context)
        elif step == 3:
            print('in step 3')
            garage_id = request.POST.get('garage_id')
            location = request.POST.get('location')
            vehicle_type = request.POST.get('vehicle_type')
            fuel_type = request.POST.get('fuel_type')
            model_number = request.POST.get('model_number')
            description = request.POST.get('description')

            print(f"Received garage_id: {type(garage_id)}")
            garage = GarageProfile.objects.filter(id=garage_id).first()
            print(garage)
            service_request = Service.objects.create(
                garage_id=garage.user.id,
                user=request.user,
                location=location,
                vehicle_type=vehicle_type,
                fuel_type=fuel_type,
                model_number=model_number,
                repair_info=description,  # Using repair_info instead of description
                status='pending',
                booking_date=timezone.now(),  # Use booking_date for the timestamp
            )
            Notification.objects.create(
                garage=garage.user,
```

```
            Notification.objects.create(
                garage=garage.user,
                message=f"New service request received from {request.user.name}.",
            )

            # Send notification to garage
            # send_notification_to_garage(garage_id, service_request.id)
            context = {
                'service_id': service_request.id,
                'garage_profile': garage,
                'rating_list': [i for i in range(garage.rating)]
            }
            return redirect('thirdStep', service_id=service_request.id)
        else:
            return redirect('bookService')
    else:
        if request.user.is_authenticated:
            garages = GarageProfile.objects.select_related('user').all()
            garage_feedbacks = []
            for garage in garages:
                feedback_count = Service.objects.filter(garage=garage.user).exclude(feedback__isnull=True).count()
                garage_feedbacks.append({
                    'garage': garage,
                    'feedback_count': feedback_count,
                    'rating_list': [i for i in range(garage.rating)]
                })

            context = {
                'garage_feedbacks': garage_feedbacks,
            }
            return render(request, 'website/bookService.html', context)
        else:
            return redirect('login_form')

```