An-Najah National University

FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

Computer Engineering Department

Software Graduation Project

Management Application for Mechanic Workshop

Students:

Omar Mahmoud Zaqout

Deyaa Bani Jaber

Supervisor:

Dr. Manar Qamhieh

Acknowledgment

To the esteemed faculty of the Computer Engineering Department, we extend our heartfelt thanks for your exceptional efforts and dedication. Your wisdom, knowledge, and unwavering commitment have greatly shaped our academic journey and personal growth.

A special and sincere thank you goes to our supervisor, Dr. Manar Qamhieh, whose dedication, guidance, and continuous encouragement have been truly invaluable. Your support has not only guided us through our academic challenges but also inspired us to strive for excellence in all aspects of our lives.

We are also deeply grateful to all those who assisted us selflessly along the way. Your collaboration, encouragement, and shared enthusiasm brought both joy and motivation, enriching our experience and driving us toward success.

Last but certainly not least, we extend our deepest gratitude to our parents. Their unwavering support, patience, and belief in us have been the cornerstone of our strength and determination throughout this journey.

This project is the result of true teamwork, and we are thankful to everyone who contributed to its success. Thank you for being a part of both our academic and personal lives.

Disclaimer

Contents

Table Figure

1 Abstract:

This project aims to develop an application for managing car repair workshops, focusing on improving workflow efficiency and offering a smart, seamless user experience through a flexible and user-friendly interface. The application also integrates artificial intelligence (AI) to assist in diagnostics and enhance the overall functionality.

The application targets four main user groups. The first is the **App Owner**, who has full administrative control over the system. They can approve or reject garage registration requests, provide technical support when needed, and monitor all registered garages. A dedicated dashboard is available for tracking system-wide statistics and performance indicators.

The **Garage Owner** submits a registration request for their garage to the App Owner. Once approved, they gain extensive privileges, including managing workshop staff, monitoring the company's budget, searching for car records using license plate numbers, and receiving repair reports from technicians. They can also broadcast general news or announcements to their employees.

Technicians play a crucial role within the system. They can submit repair reports and continuously document the condition of vehicles, contributing to organized and traceable operations. Additionally, technicians can use the AI-powered diagnostic tool embedded in the application to analyze car issues based on the symptoms provided, helping them reach accurate and fast diagnoses.

Customers benefit from the application in various ways. They can view their vehicle's repair history, submit help requests in case of breakdowns, and get a list of nearby garages ranked from nearest to farthest. They can also directly chat with the garage that accepts their request. Moreover, customers can use the built-in AI tool to analyze problems by entering visible symptoms or dashboard indicators such as the "Check Engine" warning, empowering them with useful insights before reaching out for service.

One of the core features of the application is the integration of artificial intelligence for fault analysis. By considering the vehicle's make, model, and provided symptoms, the system can offer accurate, fast preliminary diagnoses. This significantly reduces repair time, improves efficiency, and enhances the overall service experience.

To enhance real-time communication and interaction among users, the system integrates **Firebase Realtime Database** along with **Firebase Cloud Messaging (FCM)** to send instant **push notifications** to both **web and mobile platforms**. These notifications are triggered when:

A new repair report is added by a technician,

A customer submits a help request,

A garage owner sends or updates a general announcement,

A garage registration request is submitted to the App Owner.

Such notifications ensure that all users remain informed, improve response time, and support seamless collaboration within the platform.

The application is developed using **Flutter** to ensure a responsive and smooth user interface across all devices. On the backend, it is built with **Node.js** and **Express.js**, utilizing **MongoDB** for data storage. The entire system is hosted on **Render**, ensuring secure, reliable, and fast access to services.

2 Introduction

In today's digital era, automotive repair workshops face growing demands for efficiency, transparency, and enhanced customer service. Traditional workshop management methods often struggle to meet these expectations, leading to disorganized operations and communication gaps. This project aims to develop a smart, AI-powered application for managing car repair workshops, offering a seamless user experience through an intuitive interface and intelligent features.

The application serves four primary user groups: the App Owner, Garage Owner, Technician, and Customer. Each user role is provided with tailored functionalities that match their specific responsibilities. By integrating artificial intelligence, the system empowers both technicians and customers to diagnose vehicle issues using visible symptoms or dashboard indicators, such as warning lights—enabling quicker, more accurate preliminary assessments.

To ensure instant communication and responsiveness, the system integrates **Firebase Realtime Database** and **Firebase Cloud Messaging (FCM)** to deliver **real-time notifications** across both web and mobile platforms. These notifications are triggered when key events occur—such as the submission of a new repair report, a help request, general announcements, or a garage registration request submitted to the App Owner. This mechanism ensures that all relevant parties are kept informed in real time, thereby improving coordination and service efficiency.

The solution is built using Flutter for the front-end, ensuring responsiveness across various devices, and Node.js with Express.js on the back-end for robust and scalable server-side operations. MongoDB is used for efficient data management, and the entire system is deployed on Render for secure and reliable service delivery.

This platform marks a significant step toward digital transformation in the automotive repair sector by automating workflows, improving communication, and enhancing the overall service experience through smart diagnostic tools and user-centered design.

3 Theoretical Background and Previous Work

This chapter presents the theoretical background related to the project, as well as a review of previous work that has been conducted in the same or related fields. It aims to provide the reader with a clear understanding of the core technologies used in the project and the context of existing solutions.

3.1 Theoretical Background

In recent years, artificial intelligence (AI) has been increasingly integrated into various industries, including automotive maintenance. One of the most promising applications of AI in this field is fault diagnosis. Traditionally, vehicle diagnostics relied heavily on manual inspection or specialized tools such as On-Board Diagnostics (OBD) systems. However, AI techniques now enable more advanced and accessible methods for detecting faults based on symptoms and contextual data.

Symptom-based diagnosis involves analyzing the signs or behaviors exhibited by the vehicle, such as unusual noises, dashboard warning lights, or changes in performance. By correlating these symptoms with common faults across different vehicle types and models, AI systems can accurately predict potential issues. This reduces the time and effort required for manual inspection and enhances the overall quality of service.

Another important concept is the **vehicle dashboard**, which displays a wide range of indicators such as engine temperature, oil pressure, battery charge level, and check engine warnings. Understanding these indicators accurately is crucial for quick and efficient fault diagnosis. Our system aims to improve this process by mapping visible indicators to possible underlying issues using a knowledge base and AI algorithms.

From a systems perspective, modern garage management requires more than just technical diagnostics. It involves managing employees, organizing workflows, monitoring performance, and interacting with customers. Smart systems integrate both technical and administrative aspects of garage operations to create a comprehensive experience for garage owners, technicians, and customers.

Our system relies on a set of modern technologies to achieve these goals, including:

- **Flutter**: A cross-platform framework for developing interactive and responsive user interfaces across multiple devices.
- **Node.js**: A backend framework used for building APIs and handling business logic.
- **MongoDB**: A NoSQL database used for storing user data, vehicle records, symptoms, and diagnostic results.
- **Cloudinary**: A specialized service for efficient image storage and management.
- **Firebase**: Used for real-time messaging and authentication.

By combining these tools, the system provides a complete solution that supports intelligent fault diagnosis and effective garage management.

3.2 Previous Work

Many systems and applications have attempted to address the challenges of vehicle diagnostics and garage management. A common approach is the use of **On-Board Diagnostics (OBD-II)** devices, which connect to the vehicle's electronic system and provide real-time data about engine performance and error codes. Applications such as **"Car Scanner"** and **"OBD Auto Doctor"** use these devices to detect faults and display related dashboard indicators. Although these tools are effective at reading standardized error codes, they often require external hardware and are limited in their ability to interpret symptoms described by users or behaviors specific to certain vehicle models.

Regarding garage management, some platforms focus on appointment scheduling, billing, and service tracking. Examples of such systems include **Shop Boss**, **Auto Fluent**, and **Garage Plug**, which offer web-based solutions to manage customer records, invoices, and service history. However, most of these platforms lack intelligent diagnostic support and do not integrate vehicle behavior analysis or dashboard indicator interpretation.

Compared to these solutions, our system introduces a more intelligent and integrated approach by combining:

- **AI-powered fault diagnosis** based on symptoms and vehicle type, without the need for OBD devices.
- A **smart interface** tailored for technicians, garage owners, and customers.
- A **cloud-based infrastructure** that supports data storage, real-time interaction, and performance monitoring.

This makes the system more accessible, especially in regions where professional diagnostic tools may not be available, and significantly enhances the user experience by unifying technical diagnostics with administrative workflow management.

4 Literature review

In the development of intelligent car repair workshop systems, several modern technologies are employed to enhance usability, automation, and scalability. This literature review presents the major technologies used in this project and references related work or documentation, following the IEEE citation style.

A. Flutter for Cross-Platform Development:
   Flutter is a UI toolkit by Google used for building cross-platform applications with native performance. It supports reactive design patterns, custom widgets, and powerful rendering capabilities, allowing developers to create consistent UIs across web and mobile platforms [1]. In the context of workshop management, Flutter enables efficient handling of user interfaces for technicians, customers, and administrators.
B. MongoDB for Flexible NoSQL Storage MongoDB is a NoSQL document-based database designed for handling dynamic, schema-less data [2]. In a car workshop management system, where data structures for vehicles, reports, and user interactions can vary, MongoDB provides a flexible backend solution. Its scalability and high performance make it suitable for both real-time and historical data queries.
C. Backend Development with Node.js and Express.js Node.js provides an event-driven, non-blocking I/O model, making it efficient for real-time applications. Express.js, built on top of Node.js, is a lightweight framework for building RESTful APIs [3]. These technologies together form a robust backend layer for managing authentication, database operations, and communication with AI services.
D. Render Platform for Hosting and Deployment Render is a cloud platform that automates the deployment of full-stack applications with built-in support for CI/CD, SSL, databases, and static hosting [4]. It simplifies deployment and ensures high availability for systems like vehicle workshop management apps.
E. Artificial Intelligence with TensorFlow Tensor Flow is an open-source platform for developing machine learning and deep learning models. It provides robust tools for training and deploying models in production environments [5]. In this project, TensorFlow was used to build a fault diagnosis model based on input symptoms and car details, aiming to reduce the time and error margin in manual diagnoses.
F. Pre-trained Object Detection with Roboflow Roboflow provides datasets, annotation tools, and pretrained models for computer vision tasks [6]. It was used to recognize dashboard symbols (e.g., warning lights), enabling users to understand car issues without technical knowledge. This enhances the application's accessibility and automation.
G. AI Applications in Automotive Diagnostics AI has been increasingly applied in automotive fault detection, using sensor data, visual recognition, or symptom analysis. Research shows that AI systems improve diagnostic accuracy and reduce repair time, particularly when combined with domain knowledge and user inputs [7], [8].

H. Real-Time Notifications using Firebase Cloud Messaging (FCM) Firebase Cloud Messaging (FCM) is a cross-platform messaging solution by Google that allows apps to send and receive push notifications and messages reliably [9]. In the context of car repair workshop systems, FCM is used to send real-time updates such as repair status, technician responses, or customer requests. Its integration into the Flutter application ensures timely communication between users and improves user engagement and satisfaction.

5 Methodology

5.1 Constraints and Earlier Work

5.1.1 Constraints and Limitations

Since the beginning of the project, we faced several realistic constraints and challenges that affected the planning and implementation process. The most notable ones include:

- **Time Constraint:** One of the major challenges was the limited timeframe, especially since programming languages like Dart and the Flutter framework were new to us. Developing a fully functional mobile application required acquiring new skills within a short period.
- **Technical Constraints (Manufacturability and Sustainability):** We used multiple technologies such as Firebase (for real-time chat), MongoDB as the main database, and Node.js for backend development. These tools were initially unfamiliar, which required extra effort to learn and implement them properly.
- **Infrastructure Constraints (Environmental):** The application requires a stable internet connection to interact with the backend API. However, the free-tier hosting on Render was relatively slow at times, which affected the system's performance during testing.
- **Image Handling Constraint (Sustainability and Manufacturability):** We encountered difficulties in efficiently storing and retrieving images. This was resolved by integrating Cloudinary, which improved the speed and reliability of image management.

These constraints required us to quickly adapt, engage in self-learning, and make well-considered decisions to ensure the success of the project within the given timeframe.

5.1.2 Standards

Our system architecture follows the Model-View-Controller (MVC) pattern to enhance code organization and maintainability. The application is divided into three main components: Model: Represents the MongoDB database that responds to requests and updates data accordingly. View:

The graphical user interface (GUI) through which users interact with the data. Controller: The Node.js backend server that manages communication between the Model and View. For better project structure, we organized the project files into folders for the UI, providers (state management), and the server, which includes controllers and routers. This modular organization facilitated flexible and scalable development (Figure 1MVC).

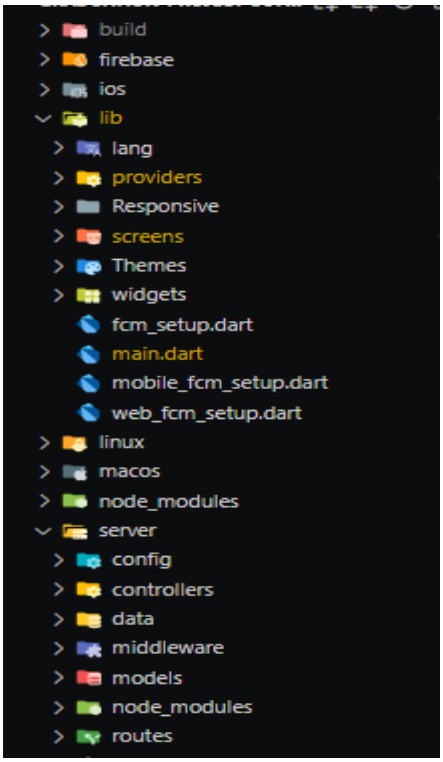


Figure 1MVC

5.1.3 Earlier Coursework

Our previous academic courses significantly contributed to the success of this project. Courses such as: Web Programming, Object-Oriented Programming, Databases, Software Engineering, provided the theoretical background and practical skills in software development and database management. Moreover, we completed several online courses covering Flutter, Dart, Firebase, and Node.js, which helped us sharpen our skills and accelerate the development process while improving code quality.

5.2 Tools, Programming Languages, and Technologies

We carefully selected the tools and technologies to ensure the development of a fully integrated and high-performance application that provides an excellent user experience. After thorough

analysis and comparison between available options, we arrived at a set of tools that enabled us to build both the frontend and backend of the application flexibly and efficiently.

5.3 Client Side

5.3.1 Design

The user interface was designed to be simple and well-organized, making it easy to navigate and ensuring a comfortable user experience. The layout was carefully customized to align with the concept of the project and meet the needs of the target users. Visually pleasing colors were selected, with a focus on text readability and smooth interaction with various elements.

5.3.2 Framework

We used Google's open-source framework **Flutter**, which enables the development of Android, iOS, Web, and Desktop applications from a single codebase. Flutter offers several major advantages: fast development with Hot Reload, high performance, beautiful UI design using reusable widgets, cost and time efficiency, excellent developer support, and an active community. It is also ideal for rapid (MVP).
To host our backend server built with Node.js, we used the **Render** platform, which allowed us to deploy our API services reliably and connect them seamlessly with the mobile application.

5.4 Programming Language

We used the Dart programming language, developed by Google specifically for web and mobile app development. Dart is a modern, easy-to-learn language that supports multiple platforms and has a syntax similar to Java, Swift, and Kotlin. Since we were already familiar with Java, this made it easier and faster for us to adapt to Dart and start developing efficiently.

5.5 Push Notifications

We relied on Firebase Cloud Messaging (FCM) to send real-time and effective push notifications to users on both the web and mobile versions of the application. FCM allowed us to deliver timely alerts and updates inside the app, enhancing user interaction and engagement.

5.6 Server-Side Framework

We used Node.js, an open-source, JavaScript-based runtime environment, to build the backend of the application. Node.js runs on the V8 JavaScript engine and is asynchronous and event-driven, making it ideal for real-time applications. Key benefits of Node.js include: Simplicity and productivity. High speed in executing operations. Multi-platform support. Easy maintenance and updates. We also used the Express.js framework with Node.js. Express is a lightweight and fast web server framework that helps manage routing and build dynamic web applications using HTML templates. It also supports asynchronous request handling, offering great flexibility in building robust APIs.

- Key benefits of Node.js include:
    - Simplicity and productivity.

- High speed in executing operations.
- Multi-platform support.
- Easy maintenance and updates.

We also used the Express.js framework with Node.js. Express is a lightweight and fast web server framework that helps manage routing and build dynamic web applications using HTML templates. It also supports asynchronous request handling, offering great flexibility in building robust APIs.

5.7 IDEs and Development Tools

5.7.1 IDE

We used Microsoft Visual Studio Code (VS Code), a lightweight and flexible code editor that supports multiple programming languages. VS Code integrates well with version control systems like Git, supports a wide range of extensions, and provides a modern development environment suitable for building contemporary applications.

5.7.2 Version Control

Since we worked collaboratively as a team, we used GitHub for version control and project coordination. We created a single repository that included both the frontend and backend, which helped us manage the project centrally and made synchronization between team members easier.

5.7.3 API Testing

We relied on Postman to test all API endpoints before integrating them with the frontend. Postman allows for executing various request types such as GET, POST, PUT, and DELETE. It provides a flexible environment for accurately and efficiently testing RESTful APIs.

5.8 Database Structure

We used MongoDB Atlas as our cloud-based database management system, integrated directly into our project using the Mongoose library in Node.js. We chose MongoDB for its NoSQL structure and JSON-like data format, which simplifies handling complex and flexible data.

- Key advantages of MongoDB Atlas include:
  I. Secure and reliable cloud data storage.
  II. Fast and efficient data access.
  III. Excellent integration with Node.js and Express.

To ensure smooth collaboration, we shared the database among team members, allowing each of us to modify, manage, and add new records easily without conflicts. Permissions and access control were implemented to maintain security.

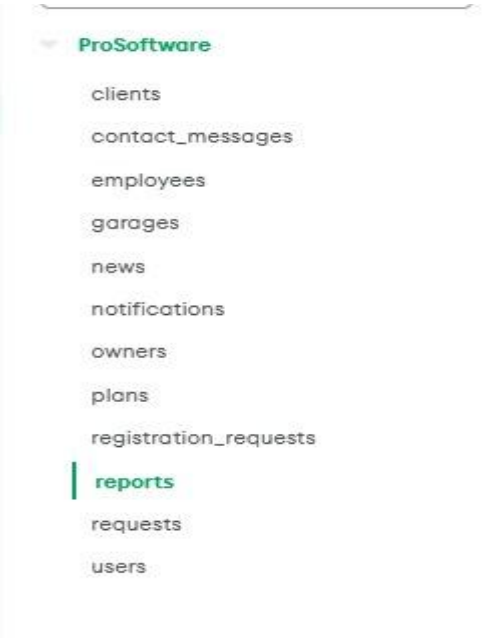


Figure 2 collections

5.9 Features of the Application

5.9.1 Implementation

Our application was developed in two versions: a mobile application and a web application. Each version was designed to meet user needs in a way that aligns with the nature of the platform. In this section, we will provide a detailed overview of the key features implemented in both versions, explaining how each feature works and the role it plays in enhancing the overall user experience.

5.9.2 Mobile Application

5.9.2.1 *Welcome Screens*

The Welcome Page is where visitors first enter the application, offering a straightforward and user-friendly design. Its two obvious options, *Log In* and *Register now*, provide a smooth experience for both new and returning users. While the Register now button enables new users to register and build their profiles, the Sign In button guides current users to log in and access their account. With a focus on simplicity of use and rapid access to the app's functionality, this page is intended to be both aesthetically pleasing and intuitive (Figure 3 welcome Page).

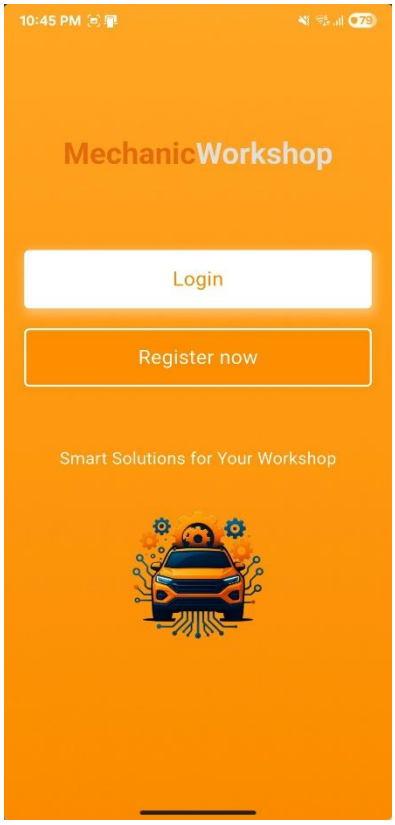


Figure 3 welcome Page

5.9.2.2 *Register Screen*

If a user does not have an account, they can create a new one by providing the necessary information. The user is required to enter a valid email address, as the application sends a verification Button to confirm their identity. After successful verification (see Figure 6complete verify).

The sign-up process differs depending on whether the user is registering as a garage owner or as a regular user, as each role requires different types of information.

- The sign-up screen includes five text fields:
    1. Full name.
    2. Email address.
    3. Phone number.
    4. Password.

When the user presses the sign-up button, the application validates the input data. The account is only created if the following conditions are met:
All fields are filled out correctly (Figure 4invalid input).
The email is not already registered, and is valid as verified by the email verification code sent to the user.
The phone number is valid and unique (if applicable in system logic).

- The password meets all of the following security requirements:

1. Minimum of 8 characters
2. At least one uppercase letter
3. At least one lowercase letter
4. At least one numeric digit
5. At least one special character (e.g., !@#\$&*~_.,-)

These validations ensure strong user authentication and enhance the security and integrity of the application. (Figure 5 valid input **Error! Reference source not found.**)

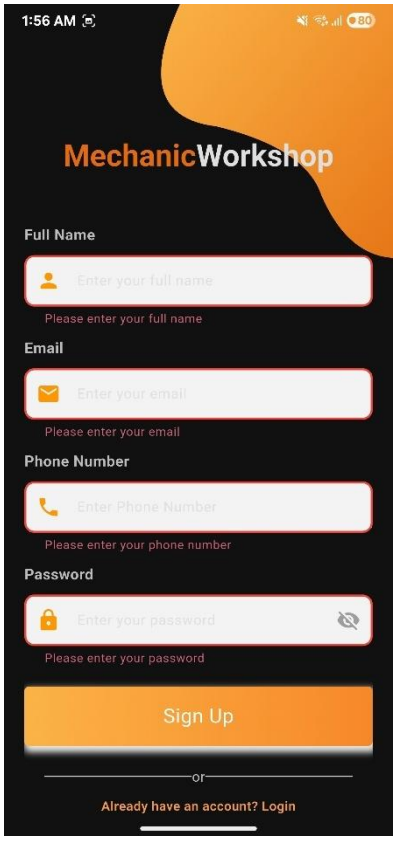


Figure 4invalid input

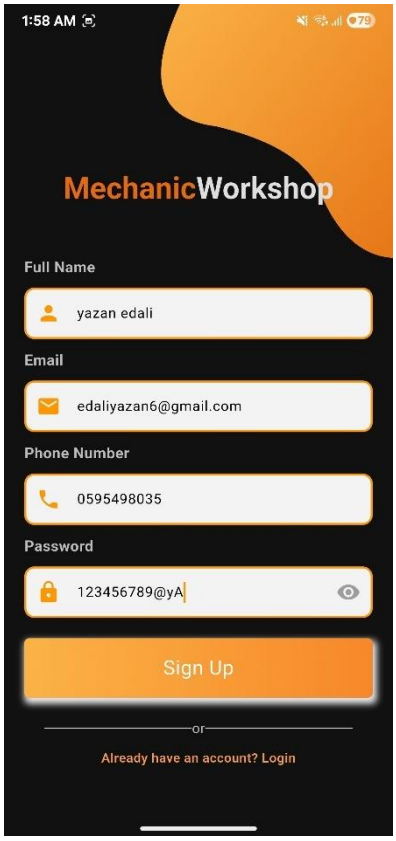


Figure 5 valid input

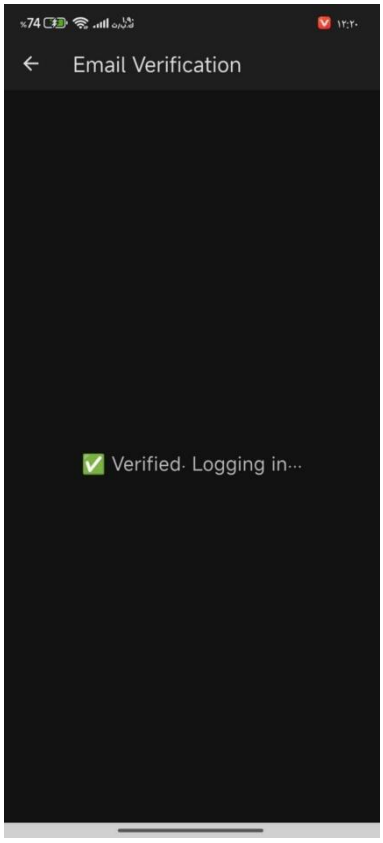


Figure 6complete verify

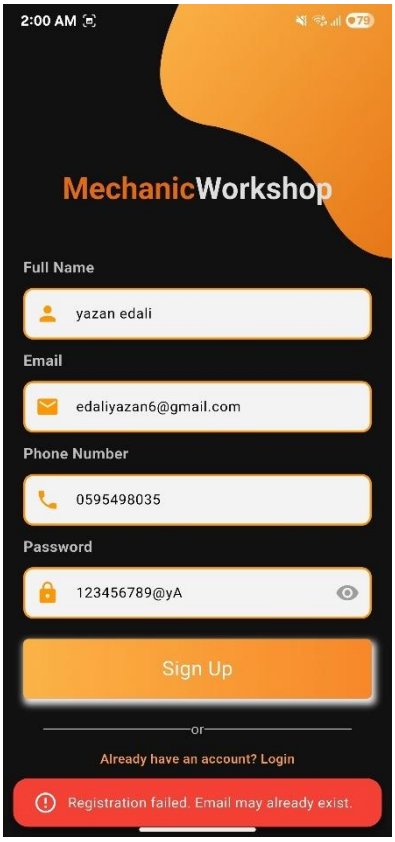


Figure 7 Email already exists

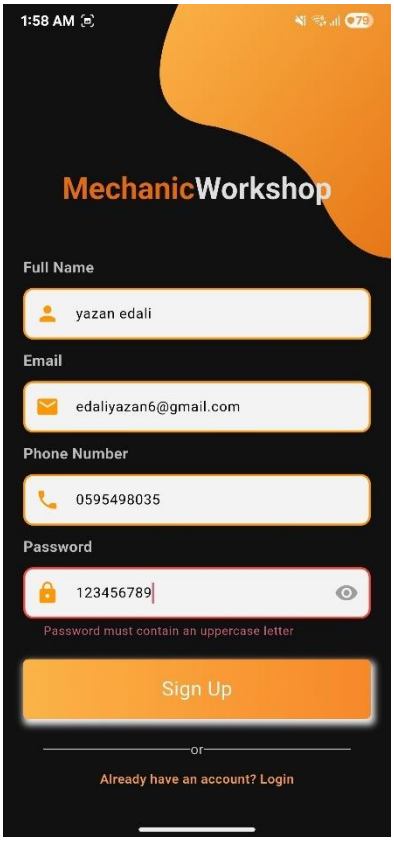


Figure 8 password weak

5.9.2.3 Email Verification Screen

After clicking the sign-up button, a verification email is sent to the entered email address. The user must confirm the email to complete the registration process. Once verified successfully, the user is automatically redirected to their designated screen based on their account type (see Figure 9screen verification).

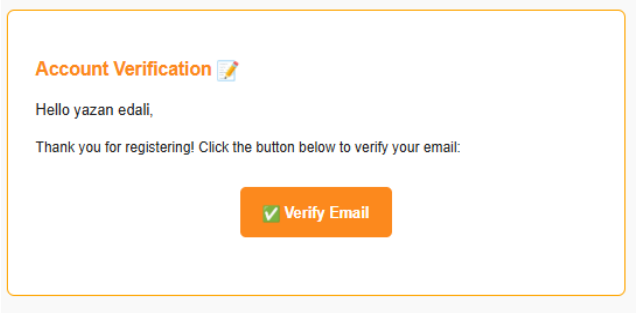


Figure 9screen verification

5.9.2.4 *Forgot Password Screen:*

This screen allows users to reset their password if they forget it. A verification email is sent to the registered email address, containing a link or code to create a new password. The screen includes fields to enter and confirm the new password, with validation to ensure both match and meet security requirements.

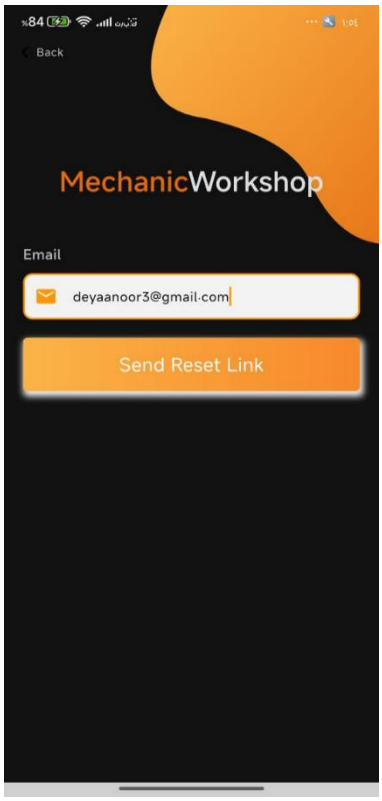


Figure 10 Reset Password

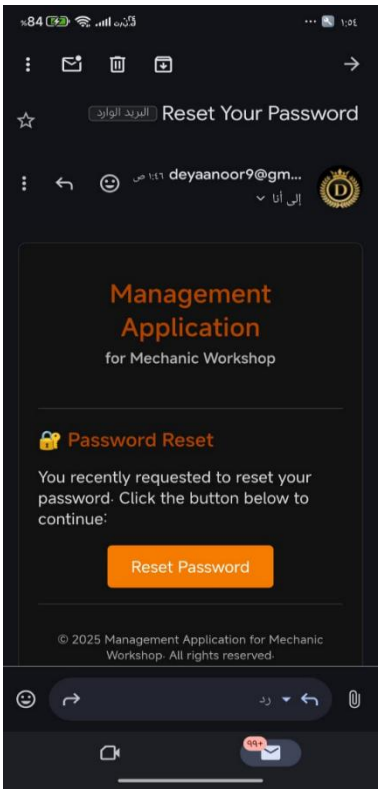


Figure 11 mail to reset password

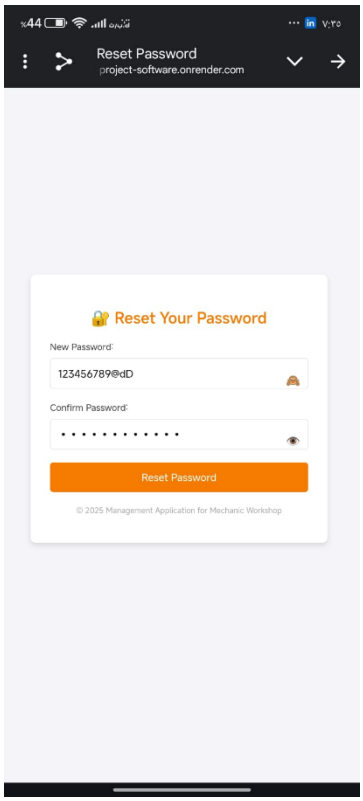


Figure 12 form reset password

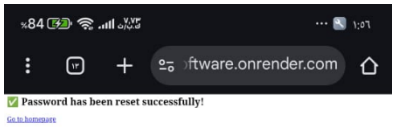


Figure 13 success reset password

5.9.2.5 Login Screen

A user needs to have an account in order to log in. If not, they can make one by going to the registration page. The user might try to log in by entering their email address and password. The application will display a warning message if any required fields are left unfilled or if the information is inaccurate. Two text fields one for the user's email address and another for their password are present on the login screen. There's also a sign in button that takes you to the home screen, but it only works when certain prerequisites are satisfied.

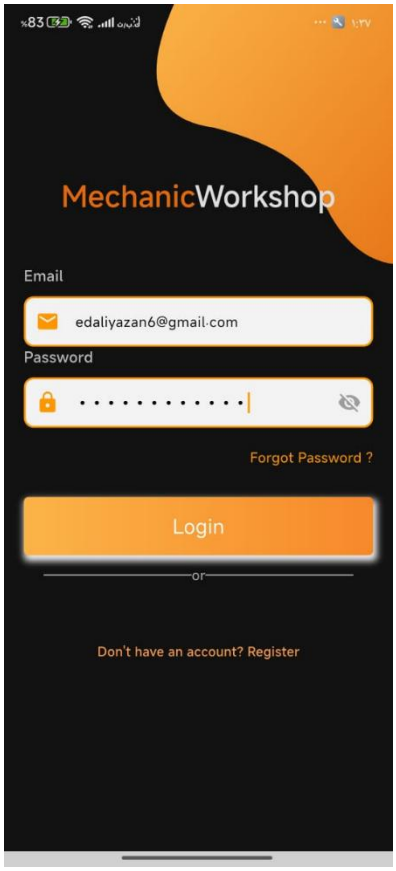


Figure 14 login

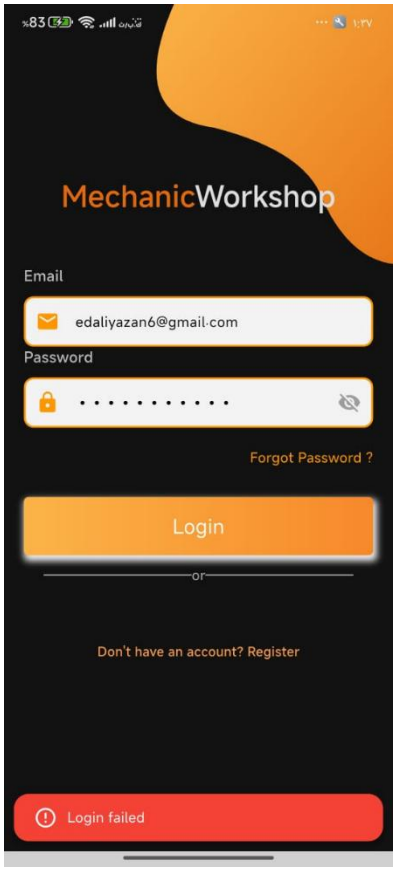


Figure 15 failed login

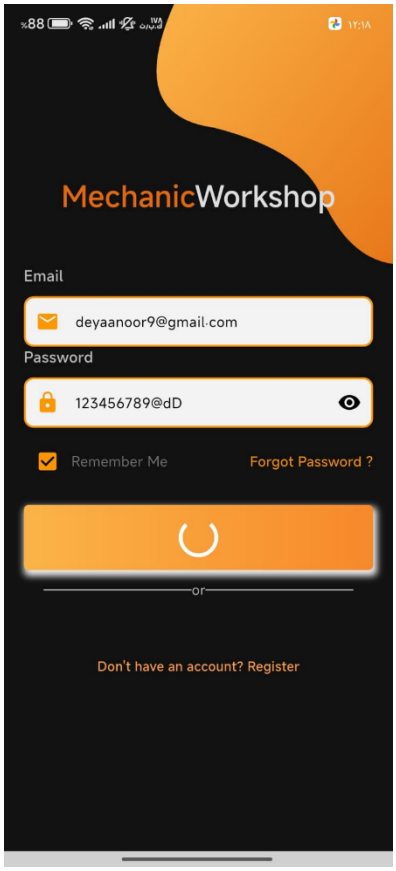


Figure 16 Login

5.9.2.6 *Admin Home Page*

- The admin home page consists of:

1. App Bar: Contains a drawer menu, profile picture, the app name, and a notification icon.
2. Drawer: Contains links to various admin pages.
3. Bottom Button Bar: Contains the main pages the admin needs.

This design facilitates quick and organized navigation for the admin to perform different tasks efficiently.

5.9.2.6.1 Dashboard Page

Contains a table listing all added garages with the owners' names and the status of each garage (active or inactive), which the admin can change. When clicking on any row in the table, garage details are displayed, including: garage name, owner's name, owner's email, garage location, phone number, subscription start date, and end date. If the location is clicked, the garage location is shown on the map. Additionally, there is a search bar that allows searching for any garage.

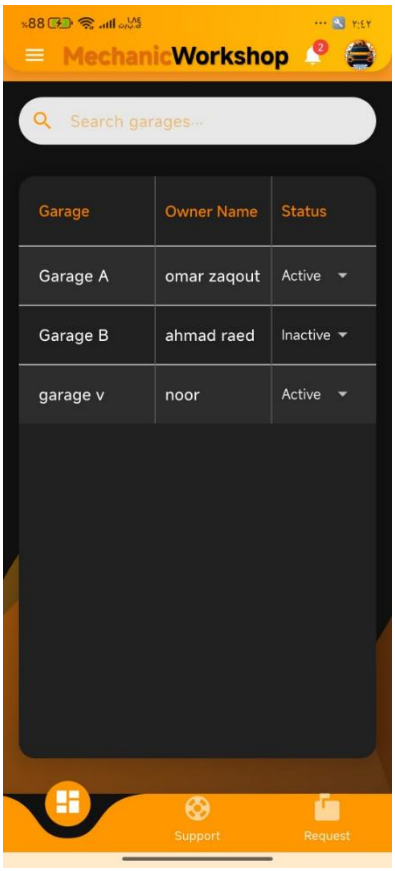


Figure 17 View All Garages

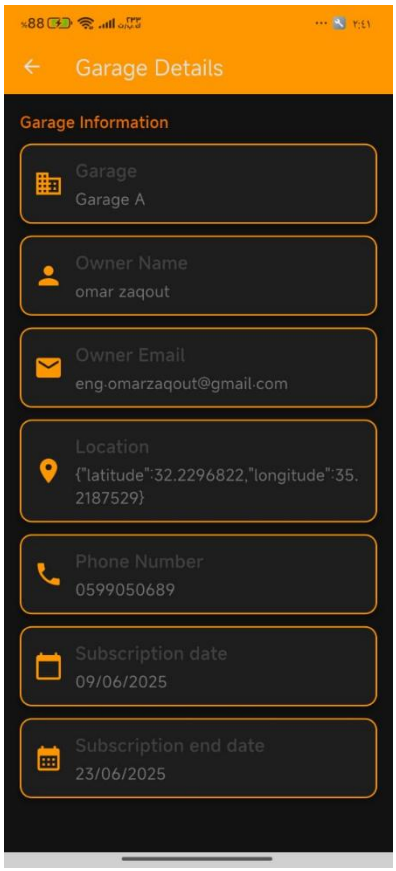


Figure 18 Garage Details

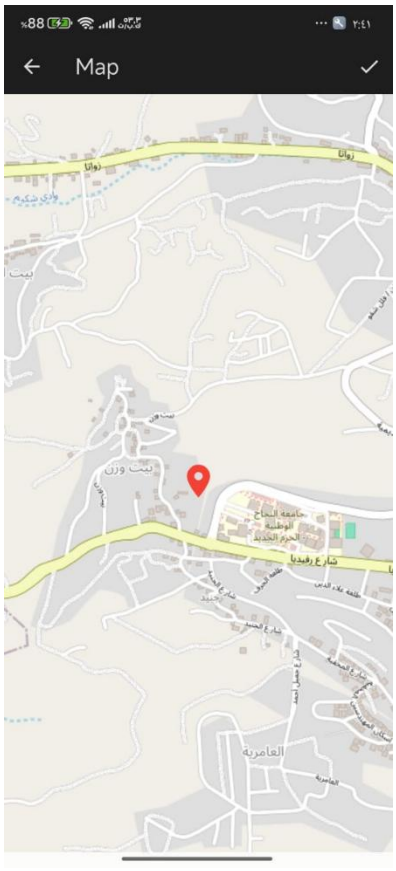


Figure 19 Location Garage

5.9.2.6.2 Support Page

The support page contains a table displaying messages sent from the "Contact Us" page by users. The table includes columns for the issue, sender, and status. Users can sort and filter the messages. Clicking any row opens a popup showing the message details, with an option to delete the message.

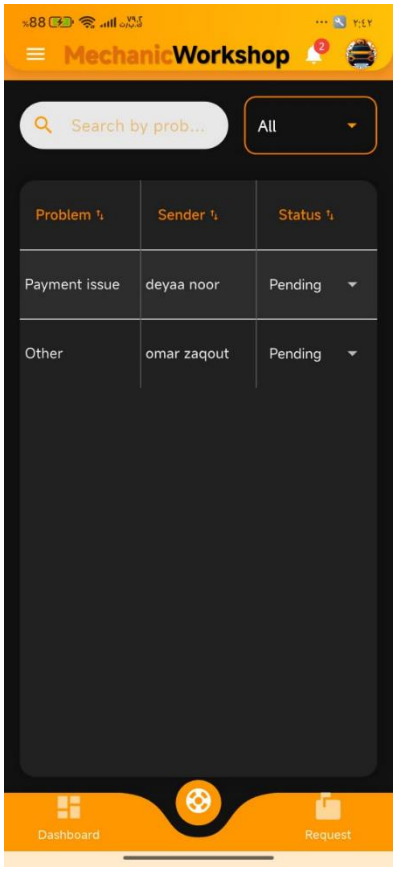


Figure 20 Support Users

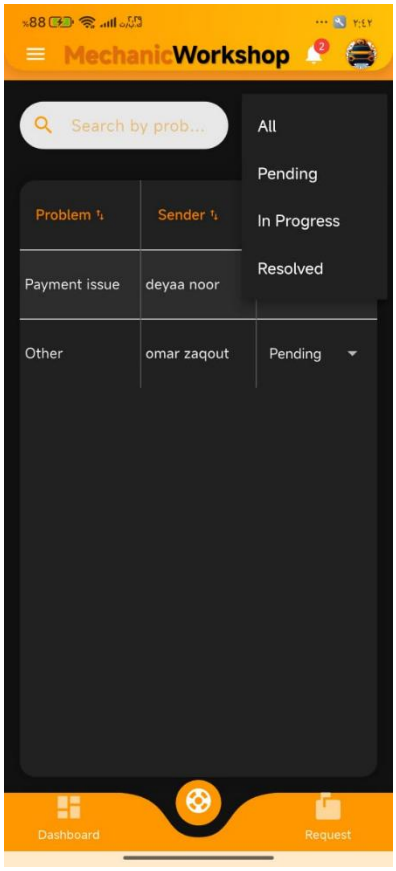


Figure 21 Filter Status

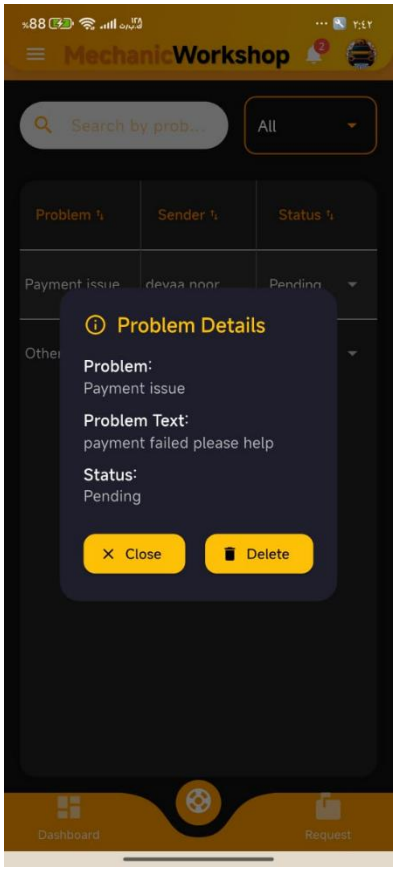


Figure 22 Problem Details

5.9.2.6.3 Request Page

The Request page contains a table listing garages that have been submitted. The admin can approve or reject each garage. Each garage shows its current status: Accepted, Rejected, or Pending. There is a search bar to search by garage name or owner's name. Filtering by status is also available.

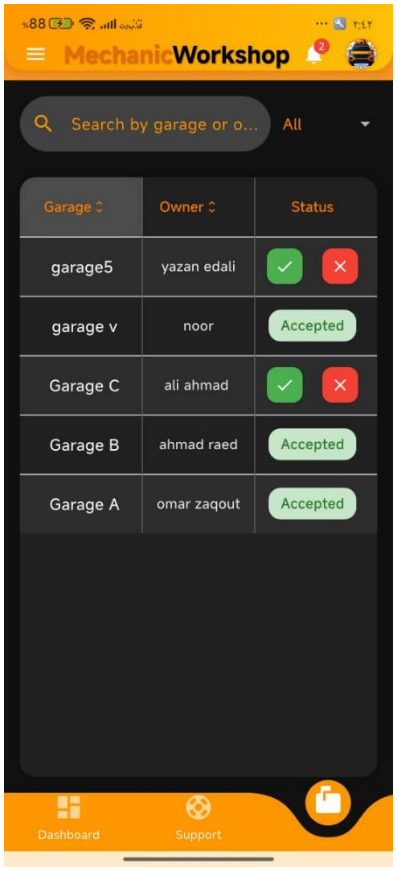


Figure 23Request Register Garage

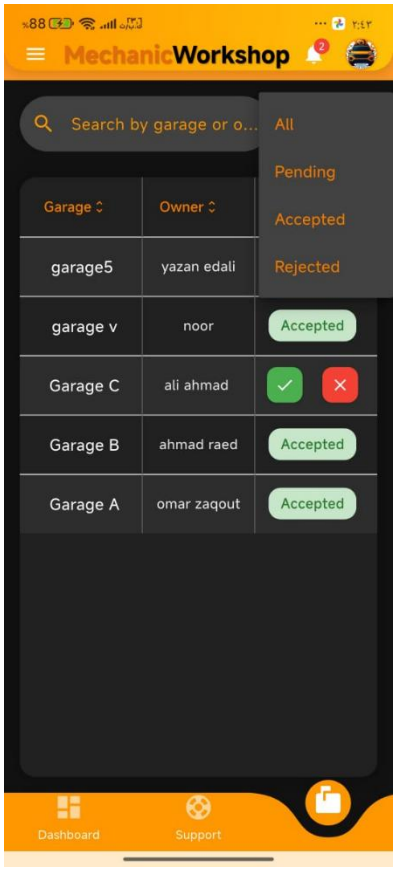


Figure 24 Filter Status

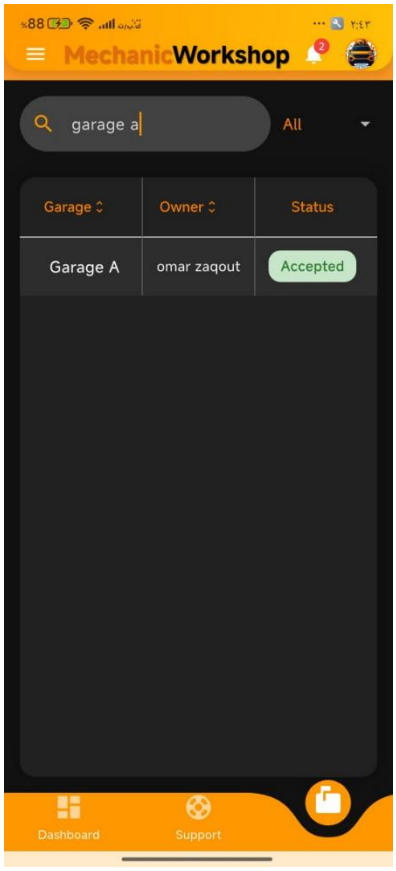


Figure 25 Search

Figure 26 Garage Registration Acceptance Email

5.9.2.6.4 Drawer

The drawer contains a header with the profile picture and email, along with links to other pages.

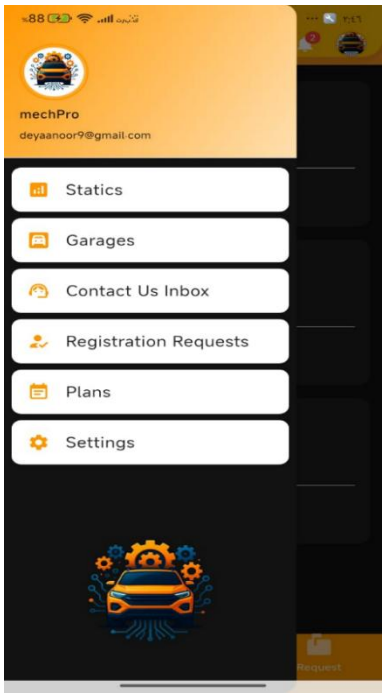


Figure 27 Drawer Admin

5.9.2.6.5 Statistics Page

This page displays key statistics such as:
Number of garages, Subscription requests, Contact messages and Trial garages.

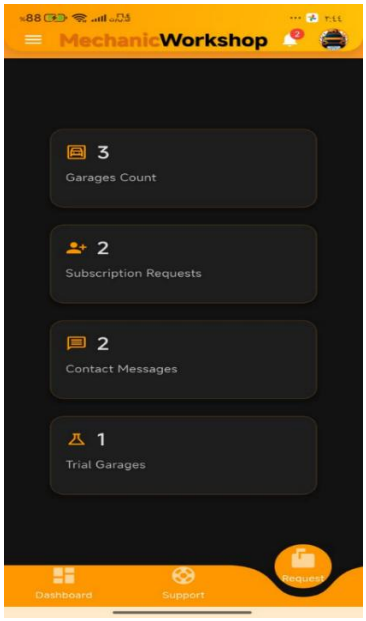


Figure 28 Statics Screen

5.9.2.6.6 Plans Page

This page displays the available subscription plans and the price of each plan. The admin can edit any plan, and after a successful update, a snack bar appears to confirm the change.

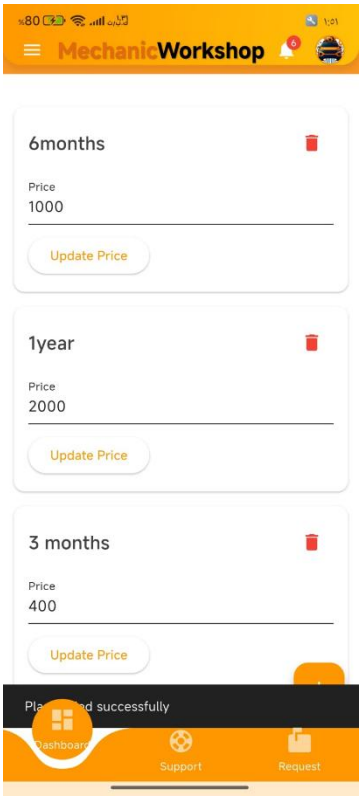


Figure 29 Plans

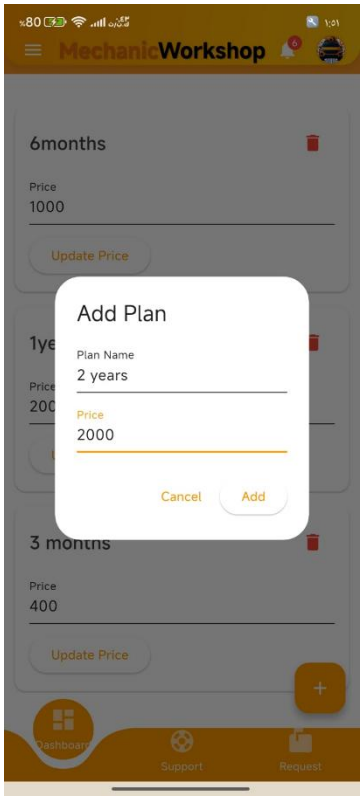


Figure 30 Add plan

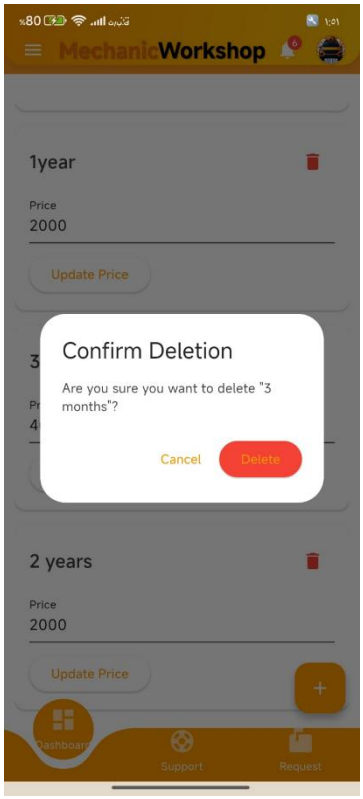


Figure 31 delete plan

5.9.2.6.7 Settings Page

This page contains the following options: Account Theme Mode: Switch between Dark and Light Mode Change Language

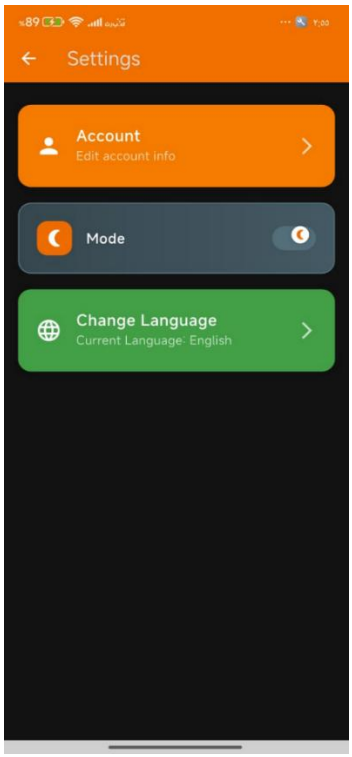


Figure 32 Settings

5.9.2.6.8 Account Page

This page displays the user's profile picture with the option to choose or capture a new one. It also includes personal information fields: Full Name Email Address Phone Number Password Each field is editable. For the password, the user can add a new password with a confirmation field.

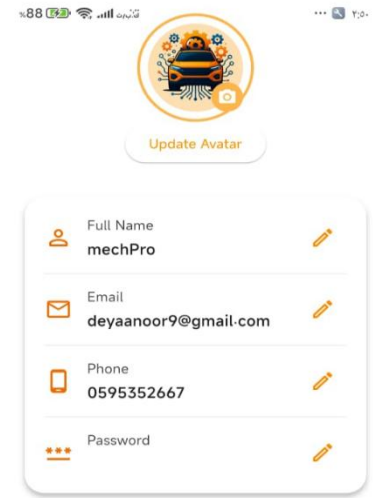


Figure 33 my Account

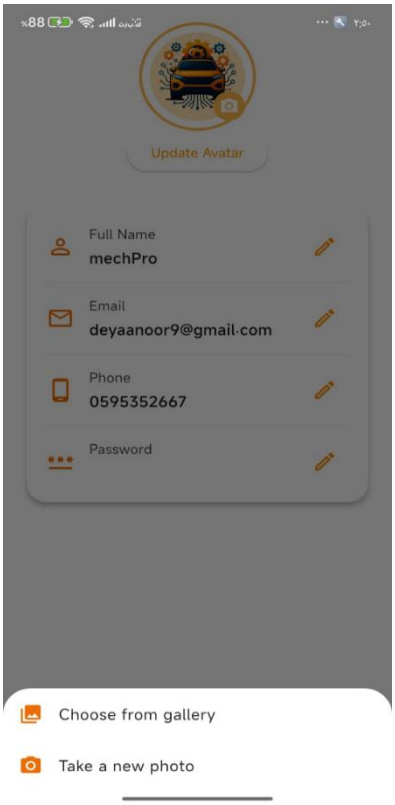


Figure 34 Update Avatar

Figure 35 Edit Full Name

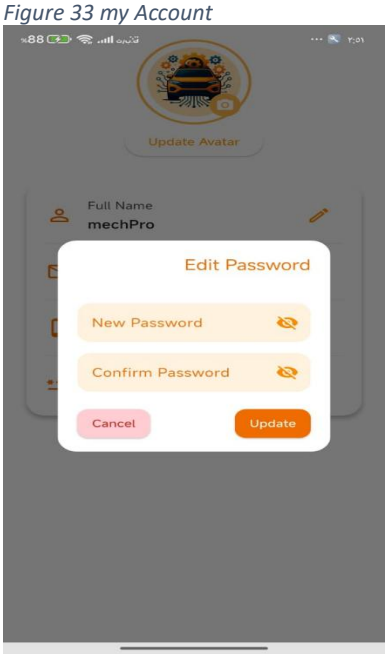


Figure 36 Update Password

5.9.2.6.9 Change Language

Page On this page, the user can change the application’s language by choosing between Arabic and English.

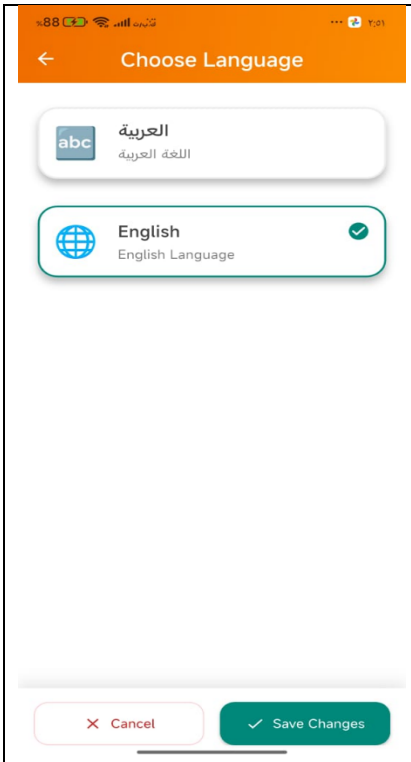


Figure 37 Select English Language

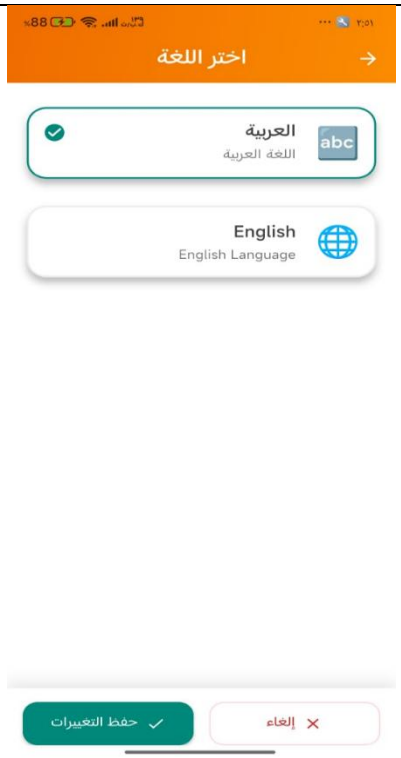


Figure 38 Select Arabic Language

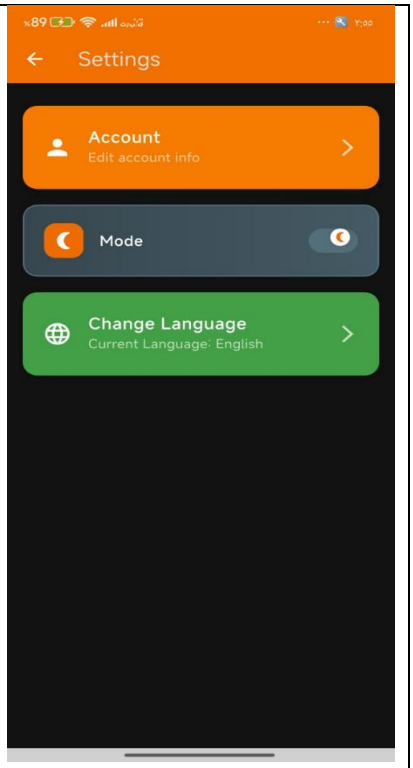


Figure 39 English View

Figure 40 Arabic View

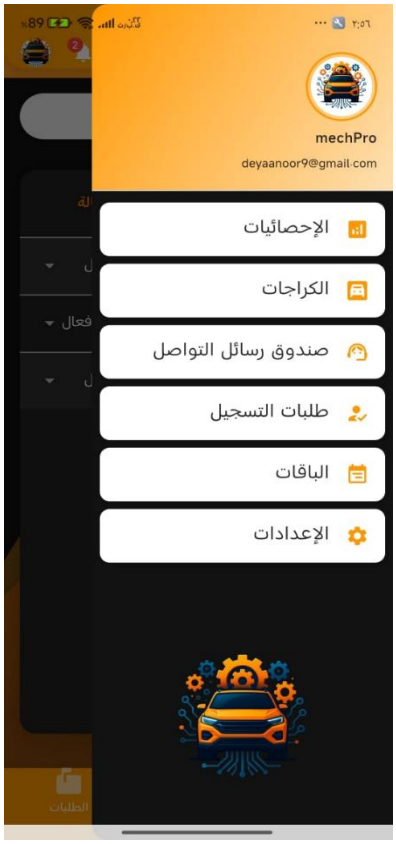


Figure 41 Arabic drawer View

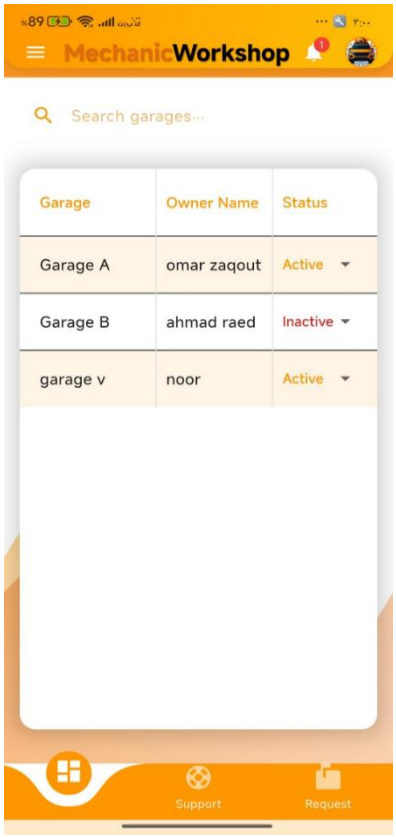


Figure 42 Garage Page English view

Figure 43 Garage Page Arabic view

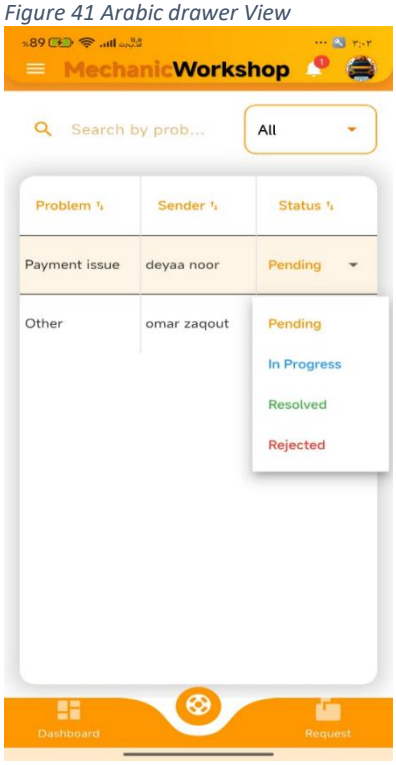


Figure 44 Support English view

Figure 45 Support Arabic View

5.9.2.6.10 Notifications Page

This page shows real-time notifications sent to the admin whenever a garage request is made. Swipe to delete: Swiping a notification removes it, after a confirmation message appears. Tap to view: Tapping a notification navigates to the Request page and clears the blue “unread” indicator.

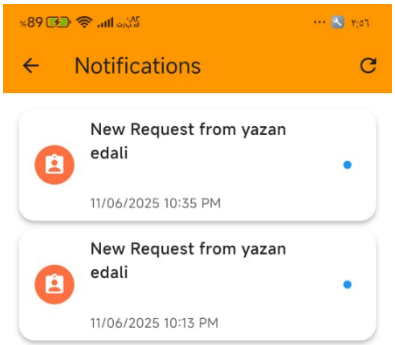


Figure 46 Notifications Screen

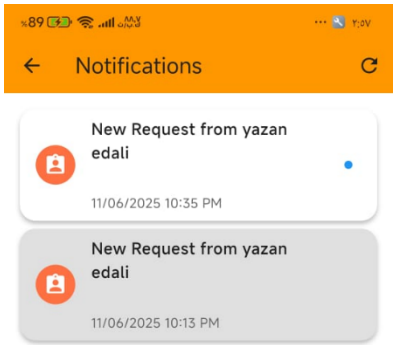


Figure 47 When click Notification

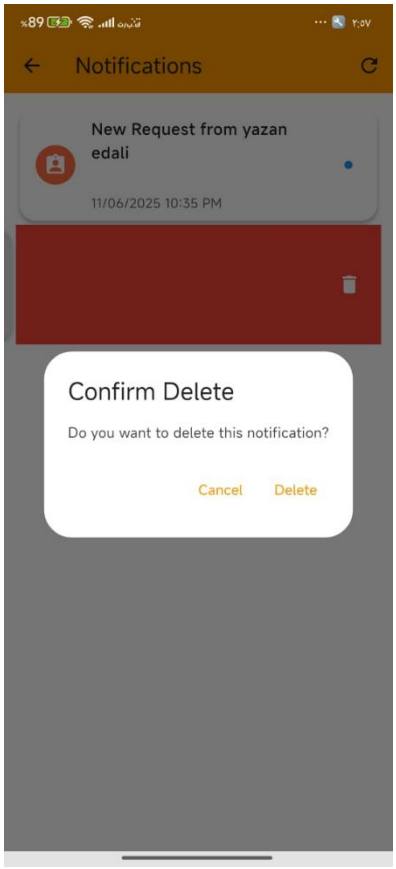


Figure 48 When swipe the notification

Figure 49 Badge notification

5.9.2.7 *Apply Request Page*

This page allows any garage owner to apply for an account in the application. The process is as follows:

1. The user first registers through the "Register Now" page and verifies their email.
2. After successful verification, they are redirected to the Apply Request page.

On this page, they:

- Enter the garage name.
- Select the location on a map (automatically detected and can be adjusted manually).
- Choose a subscription type:
    a. Trial
    b. 6 Months
    c. Yearly
- The selected subscription's **price** is displayed.
- The user proceeds to the **payment section**.
- Upon confirmation, the user is **redirected to a secure Stripe payment page**.
- After a successful payment:
    - A **Snack bar** confirms the request was submitted.
    - A **real-time notification** is sent to the admin.
- The request status is marked as **Pending** until reviewed.
- Once the request is either **Accepted** or **Rejected**, the user receives an **email notification**.

If the user tries to re-access the Apply Request page while their request is still in pending status (not yet accepted or rejected), a dedicated page is displayed with the message:

- Your request is being processed
- Your request is under review by the administration.
- Please wait. If accepted or rejected, you will receive an email.

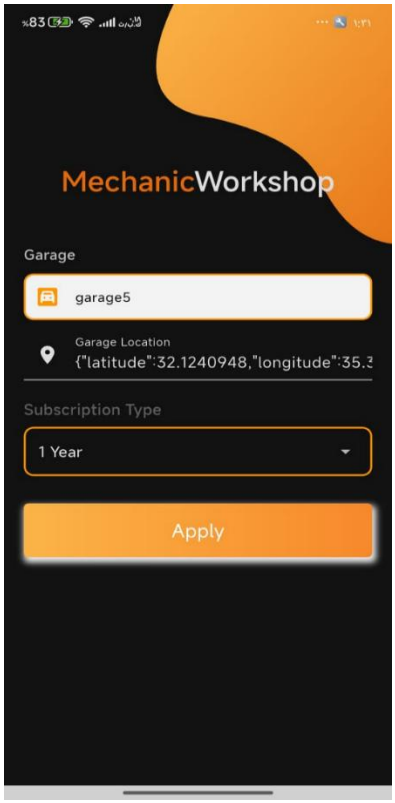


Figure 50 form request

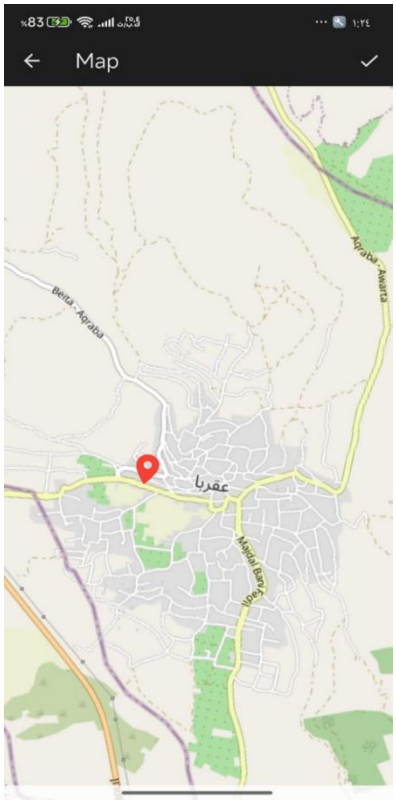


Figure 51 current location

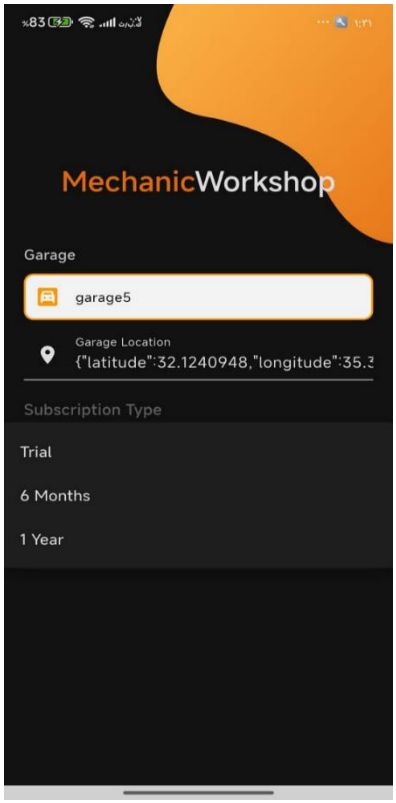


Figure 52 select type sub

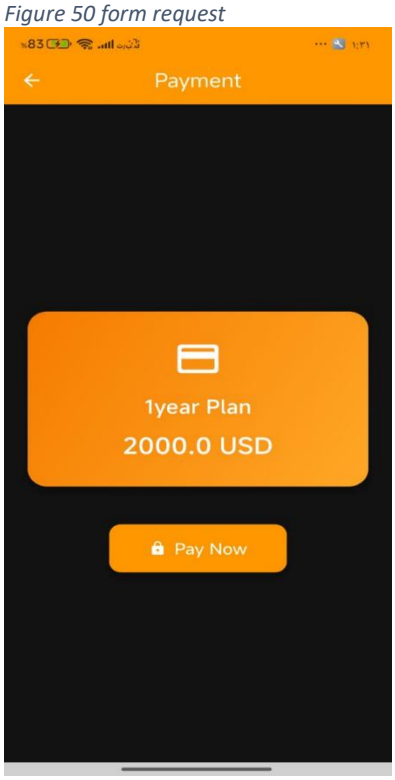


Figure 53 Payment

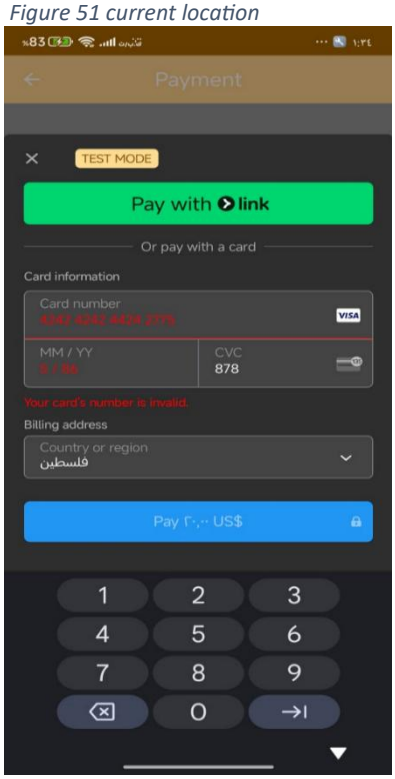


Figure 54 Invalid input form

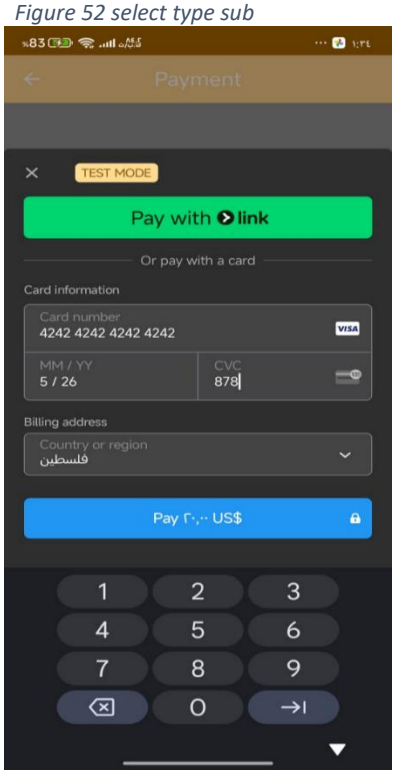


Figure 55 Strip Form

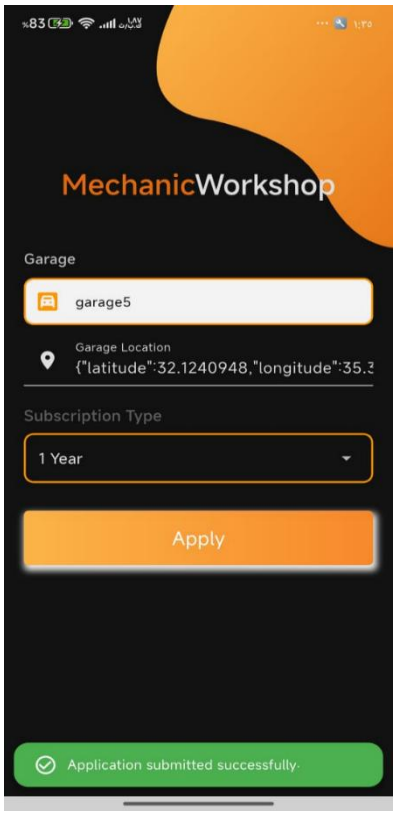


Figure 56 Apply Request Garage

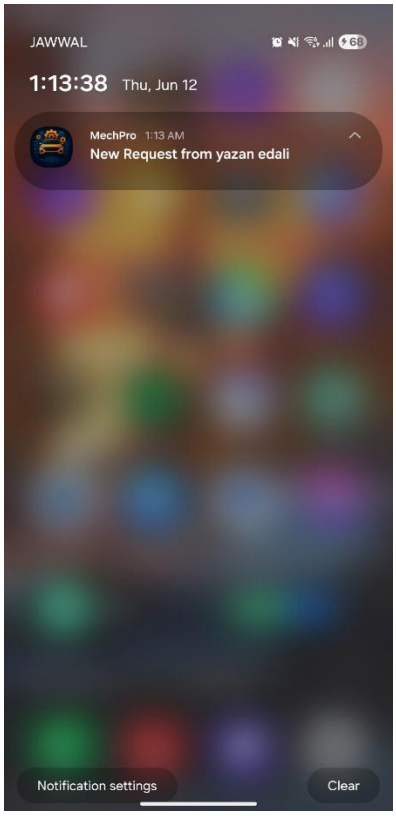


Figure 57 Notification Add Report

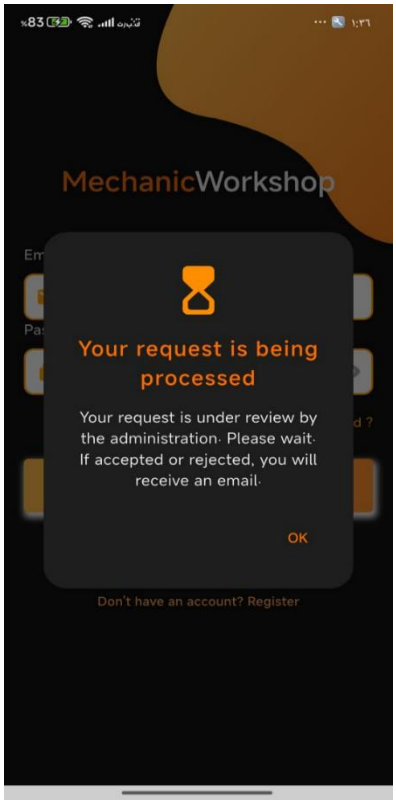


Figure 58 When a request is pending

5.9.2.8 Owner Pages

5.9.2.8.1 Home Page

The home page for the owner includes:

- App Bar: Contains the profile picture, notification icon, and a drawer menu.
- Bottom Navigation Bar: Includes the main pages relevant to the owner.
- Drawer: Contains links to the additional/side pages.
- Dashboard Page The dashboard provides key insights and analytics related to garage activity. It includes:
    - Monthly Statistics:
        - Number of car repairs.
        - Number of employees.
        - Total salaries.
        - Total cost.
    - Pie Chart:
        - Shows the distribution of the top 5 most repaired car models in the garage.
        - Bar Chart: Displays the top 5 most active employees based on the number of reports submitted.
        - Each bar represents an employee's name and the count of reports.

- Recent Repairs Table: Shows a table of the latest 5 repairs done in the garage.

- Bar Chart:
  Displays the top 5 most active employees based on the number of reports submitted.
  Each bar represents an employee's name and the count of reports.
- Recent Repairs Table:
  Shows a table of the latest 5 repairs done in the garage.

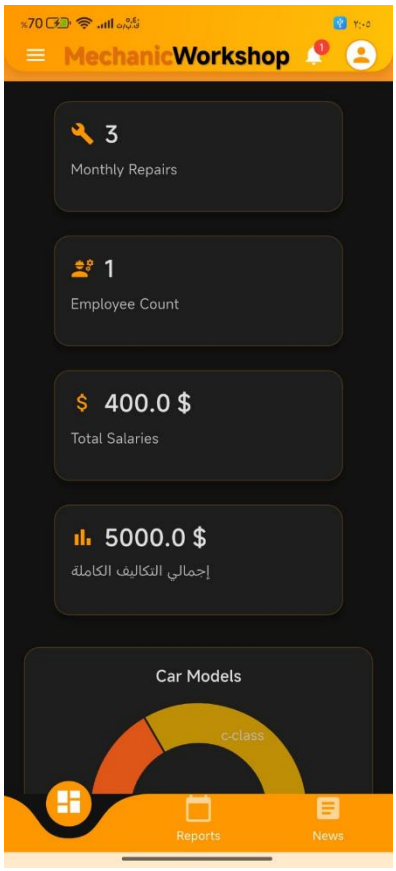


Figure 59 Dashboard Screen

Figure 60 cont. Dashboard

Figure 61 Dashboard Repairs Table

5.9.2.8.2 Reports Page

- Displays a table of all repair reports.
- Includes sorting by car plate number, car owner's name, or repair date.
- Allows searching by car owner's name or plate number.
- Clicking on a report opens its detailed view, with options to:
  - Edit or delete the report (only if the user is the owner).
  - Download or print the report as a PDF using the icon in the App Bar.
  - A floating action button at the bottom of the dashboard allows the user to add a new report.

- Upon clicking "Add Report", a page opens with:
    - A searchable list of previously repaired vehicles (auto-fills personal info if the car was serviced before).
    - An option to create a new report using the "Add Report" button.

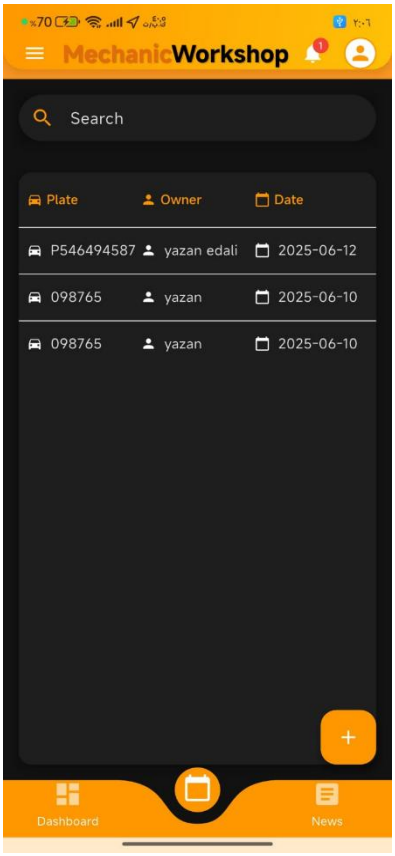


Figure 62 Reports

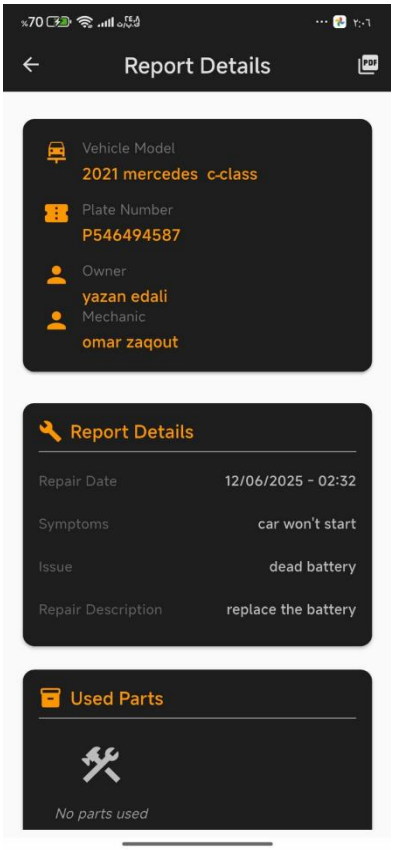


Figure 63 Report Details

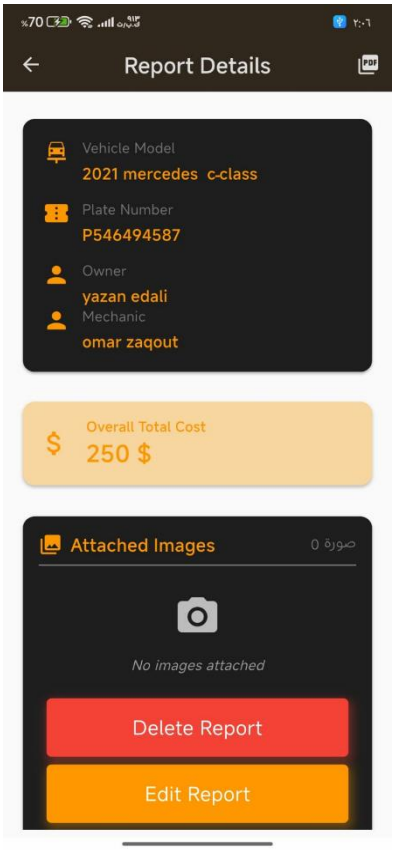


Figure 64 cont. Repairs Table

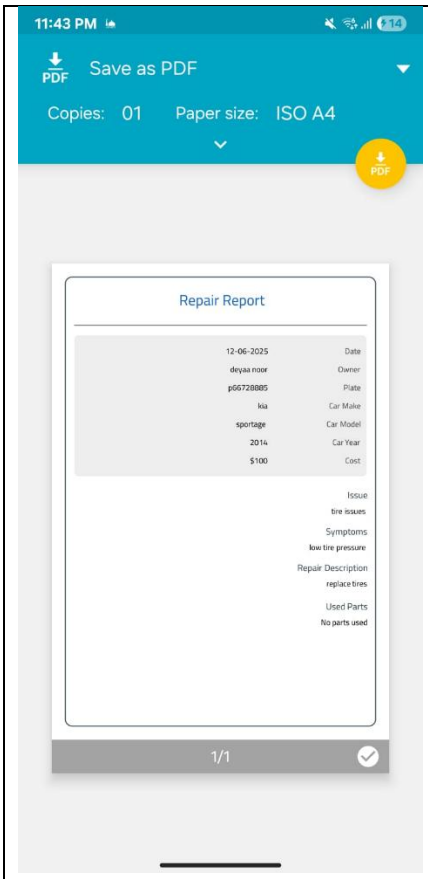


Figure 65 Export Pdf Report

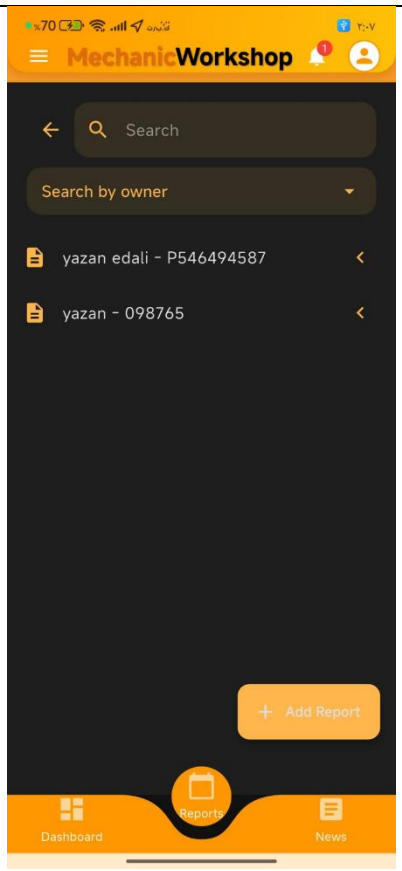


Figure 66 Report car owner is already registered

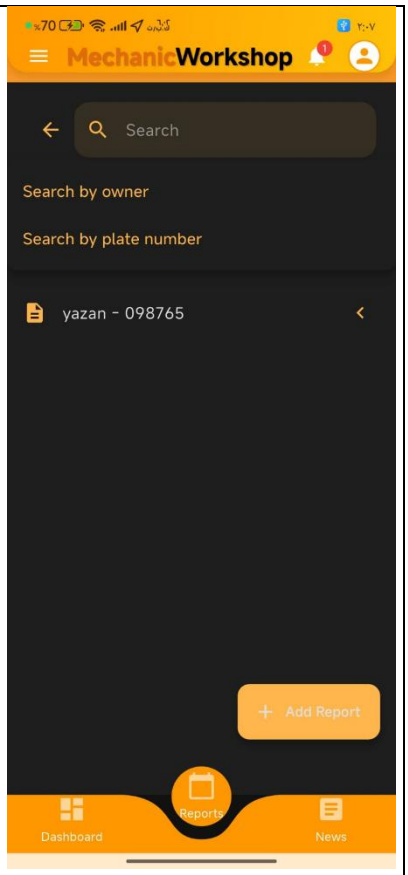


Figure 67 search Report

5.9.2.8.3 Add Report Page

- Includes the following fields for entering repair details:
  - Car owner's name
  - Car plate number
  - Cost
  - Car makes
  - Car model
  - Car year
  - Problem title
  - Car symptoms
  - Used parts Repair
  - Description
  - Option to add images

- AI Integration This page is integrated with a custom-trained AI model (built using Python and TensorFlow) to analyze and suggest repair solutions.
    - Inputs to the model:
        - Car makes
        - Car model
        - Car year
        - Problem title
        - Car symptoms
- By clicking "AI Analysis", the inputs are sent to the model, which analyzes them and suggests a repair solution by filling in the repair description field automatically.
- Voice Input A voice recording feature is available in the repair description field to simplify long explanations.
- Submission
    - On clicking Send:
        - The report is stored and added to the Reports Page.
        - A popup confirms success or failure.
        - A notification is sent to the owner, since both owners and employees can submit reports.

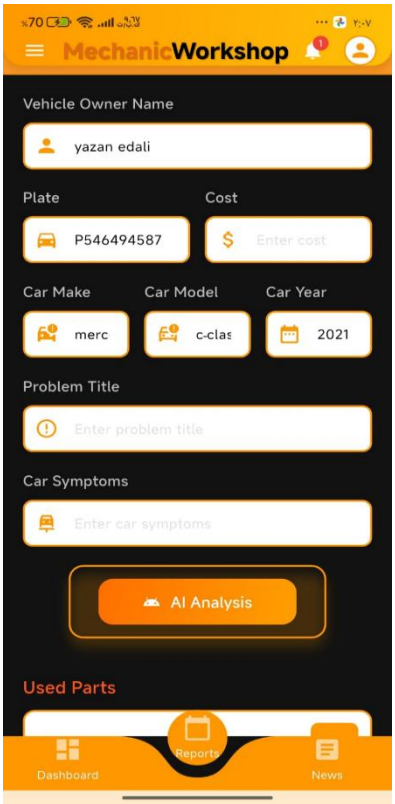


Figure 68 add report of exist client

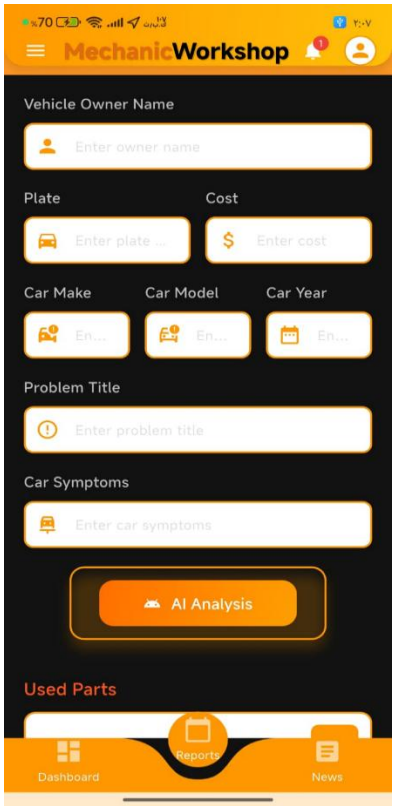


Figure 69 add new report of new client

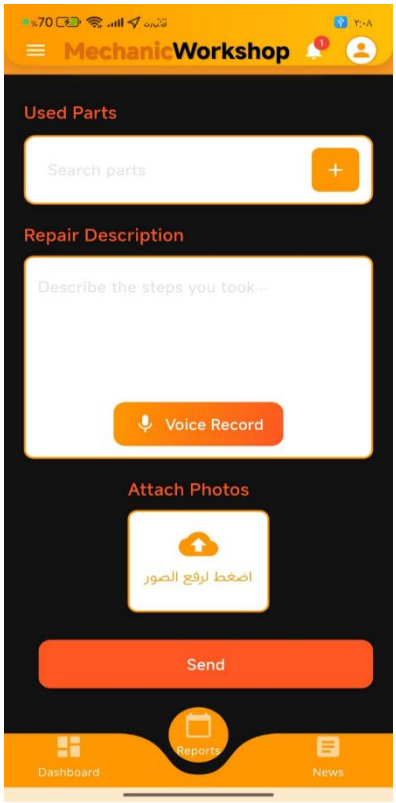


Figure 70 cont. new report

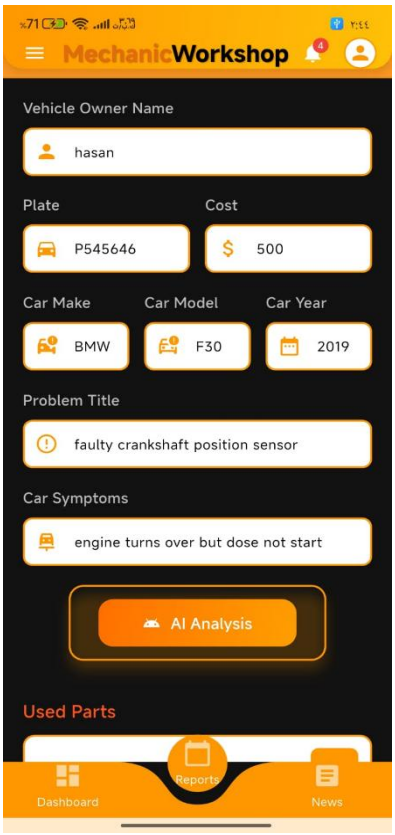


Figure 71 add new report

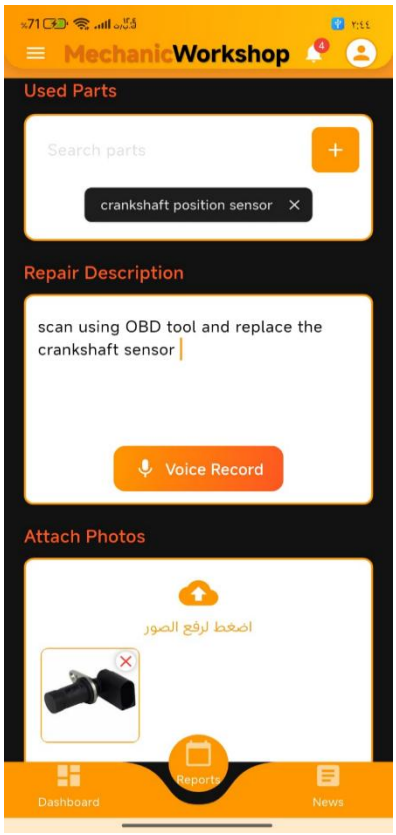


Figure 72cont. adds new report

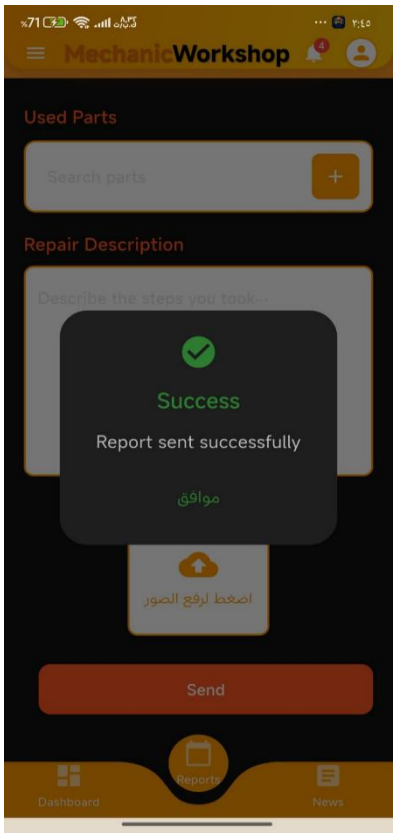


Figure 73 send new report is success

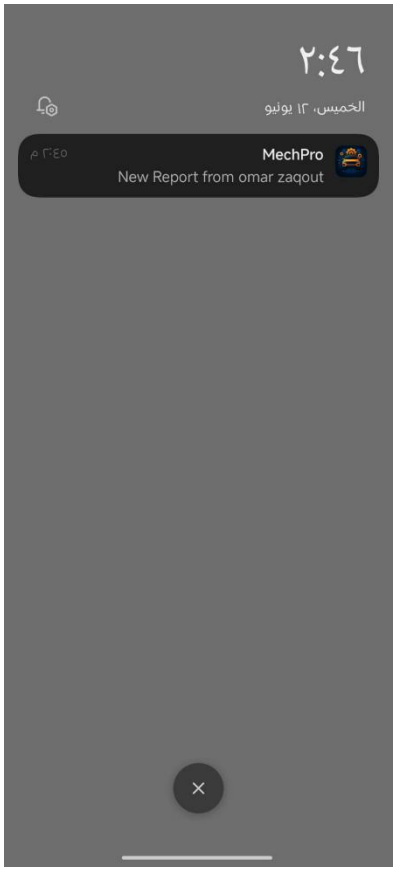


Figure 74 notification owner

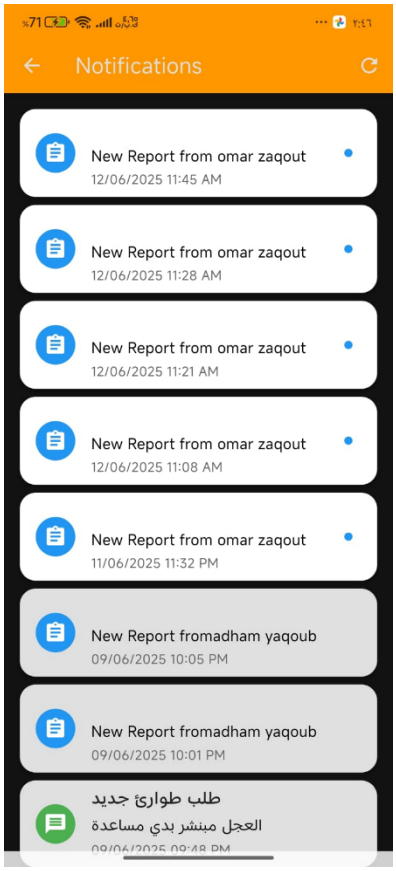


Figure 75 notifications screen

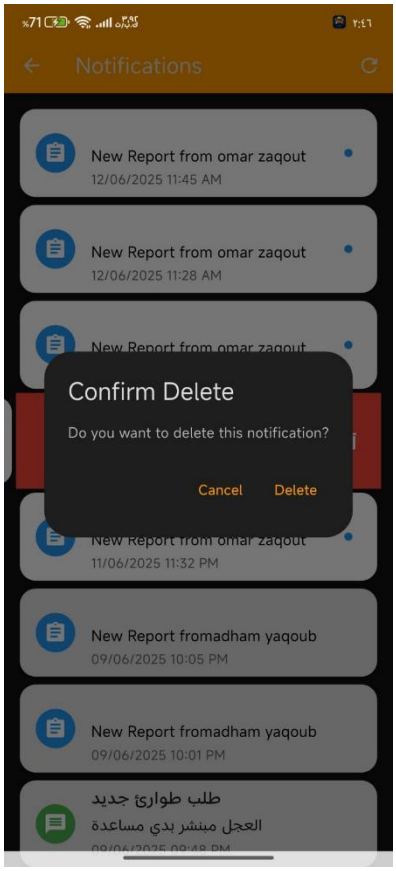


Figure 76 delete notification

5.9.2.8.4 Instant News Page

- Displays all announcements and news published by the garage owner.
- These announcements are visible to employees as well.
- Each announcement includes:
    - Date of publication
    - Title Content
- The owner can edit any published announcement.
- Upon publishing or updating a news item, an instant notification is sent to all employees.

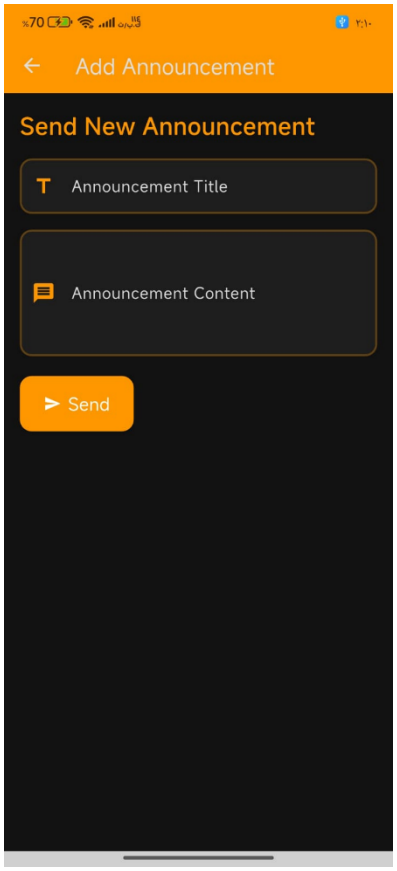


Figure 77 form add news

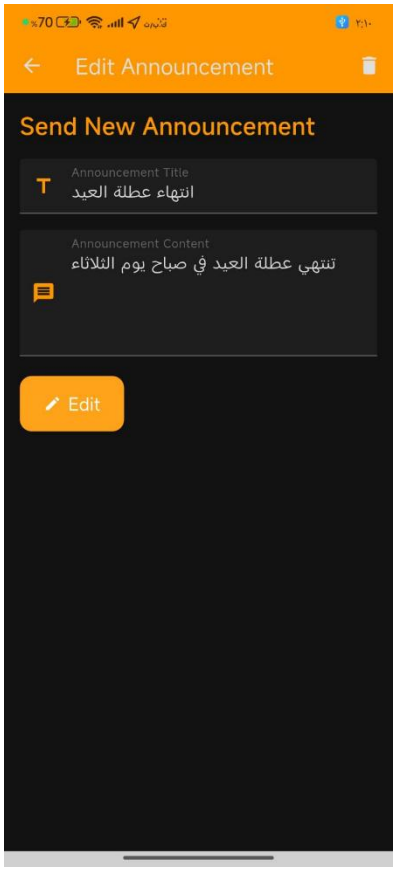


Figure 78 update news

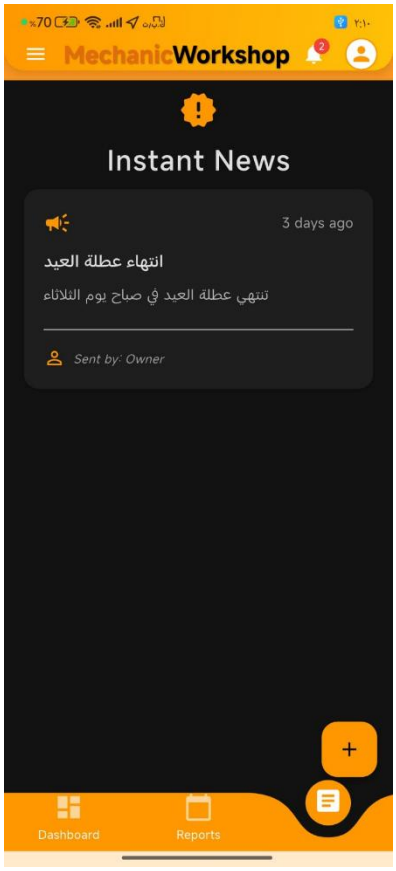


Figure 79 News Screen

5.9.2.8.5 Employees Page

Contains a table listing all employees associated with the garage.

- The owner can:
  - Add new employees
  - Edit existing employee data
  - Delete employees
  - Includes a search bar to find employees quickly.

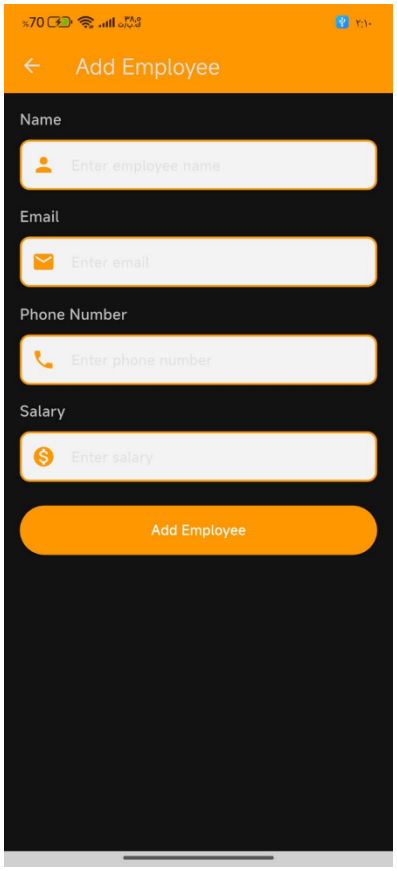


Figure 80 add employee screen

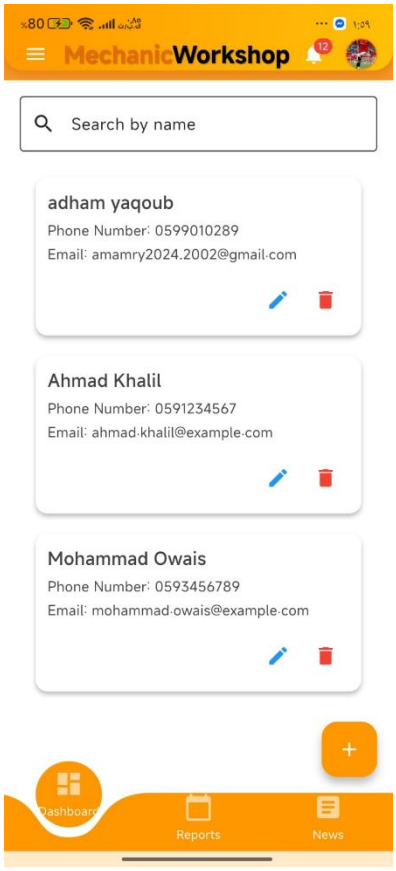


Figure 81 view all employees

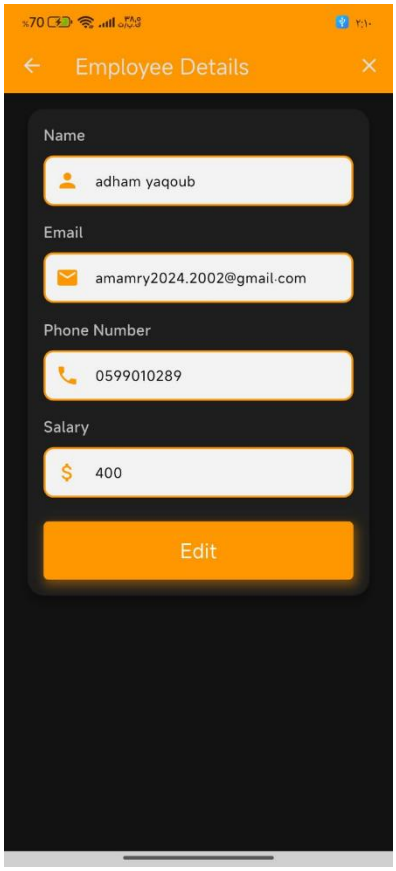


Figure 82 employee details

5.9.2.8.6 Clients Page

Displays a table of all clients linked to the garage.

- The owner can:
  - Add new clients
  - Edit client information
  - Delete clients
  - A search bar is available to filter or locate clients easily

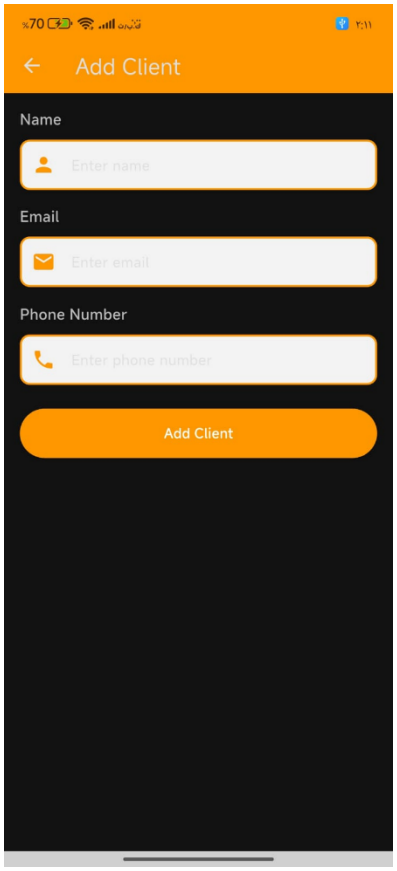


Figure 83 Add Client

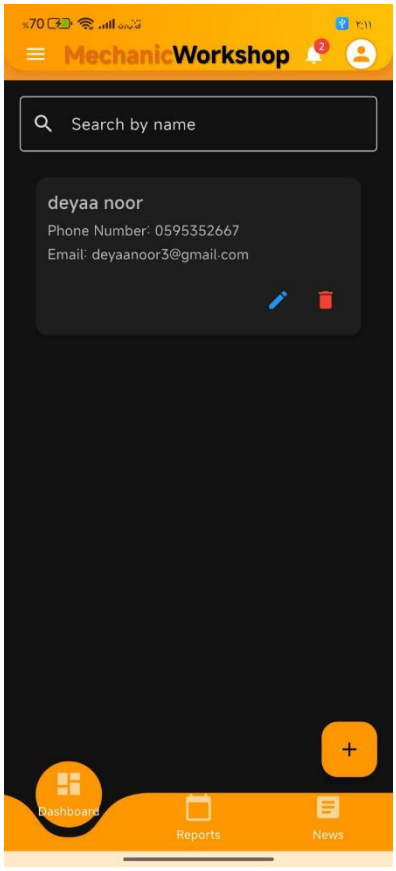


Figure 84 view clients

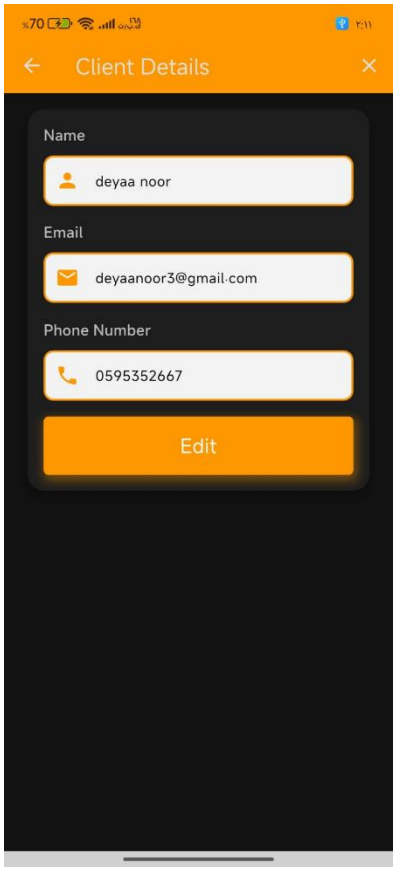


Figure 85 client details

5.9.2.8.7 Requests Page

This page displays all assistance requests submitted by clients.

- When selecting a request:
  - The conversation history appears.
  - The garage location and the client's location are shown on the map.
  - The owner has the ability to:
    - Delete the request.
    - Update the request status as needed.

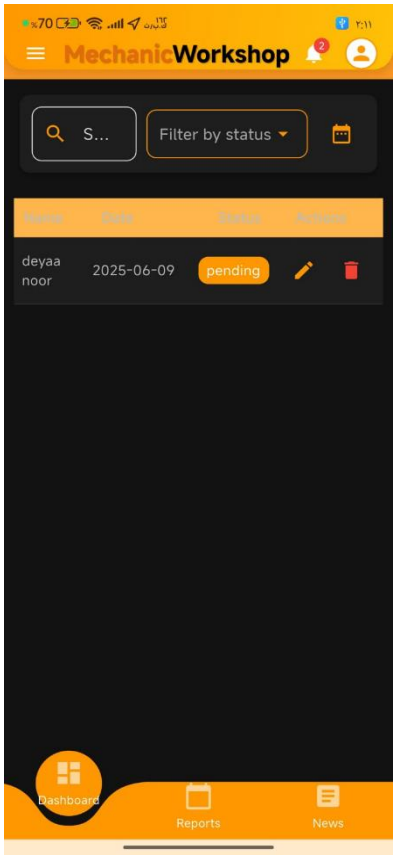


Figure 86 View Emergency Request

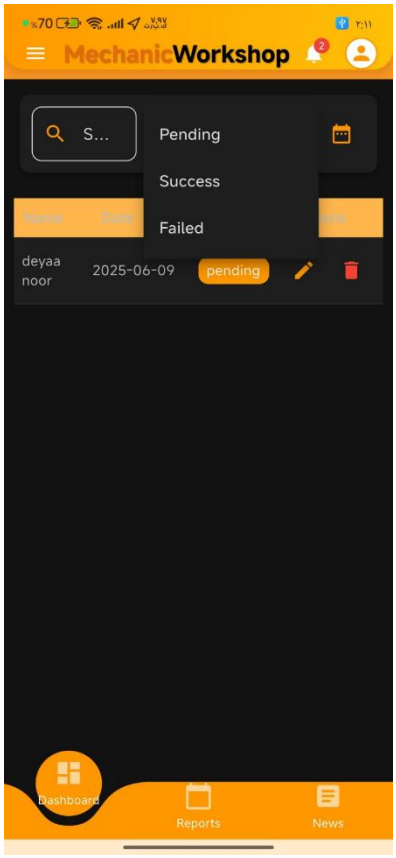


Figure 87 Filter Status

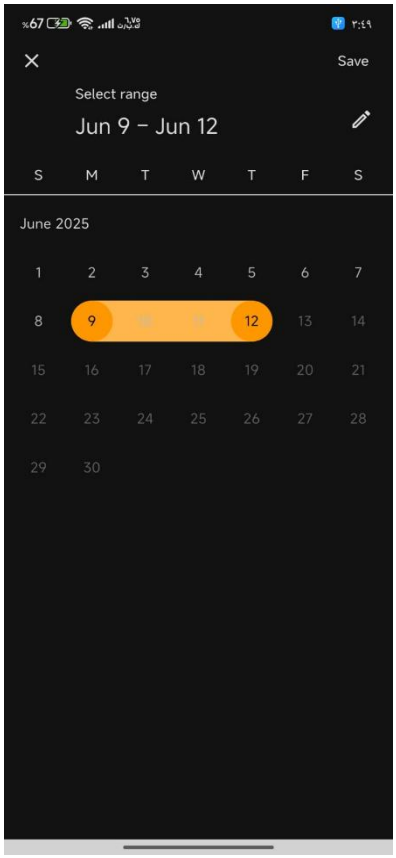


Figure 88 Filter using date

Figure 89 conversation client

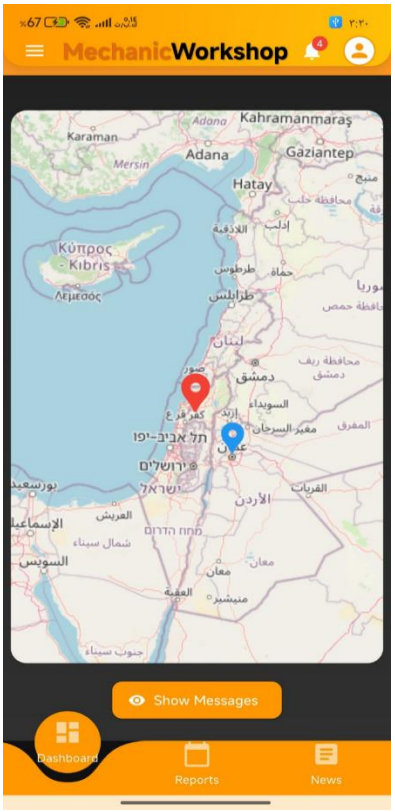


Figure 90 location garage and client

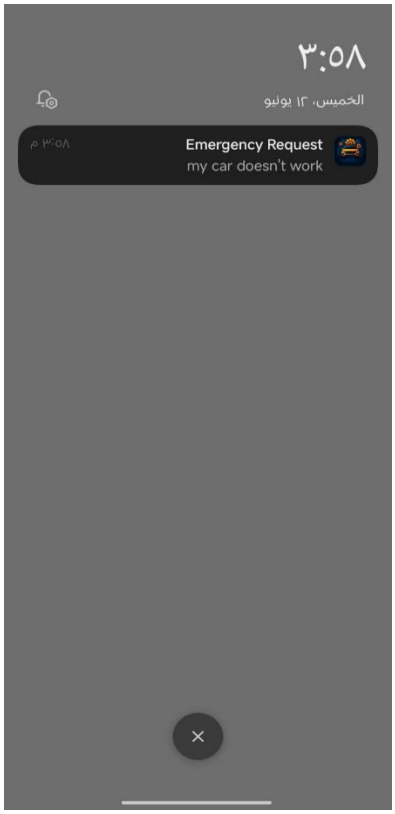


Figure 91 notification emergency request

5.9.2.8.8 Subscription Page

This page allows the garage owner to renew their subscription.

- It displays:
  - Subscription details
  - Start and end dates
  - Current status (Active or Inactive) The owner can tap the "Renew Subscription" button to view available plans and select one.
  - After choosing a plan, they are directed to the payment page to enter their details.
  - Stripe is used to verify and process the payment.
  - Upon successful payment, a snack bar confirmation appears and the subscription is extended.
  - Additional Features:
    - 15 days before expiration, an email reminder is sent daily to the garage owner.
    - If the subscription expires:
      - The owner is redirected to the Subscription page upon login.
      - Employees see a popup message indicating the garage's subscription has expired.

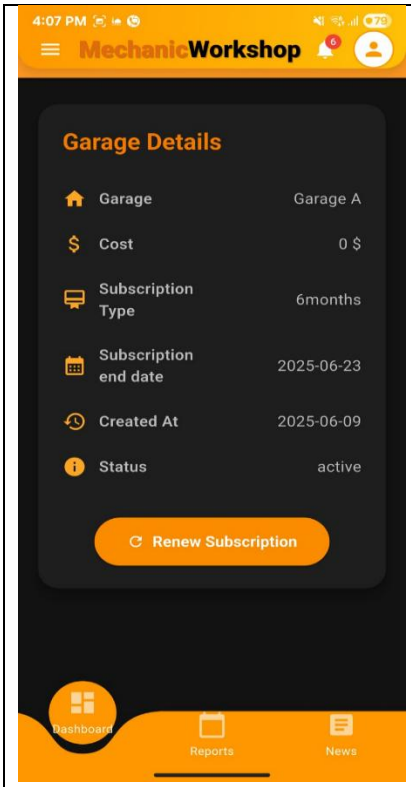


Figure 92 Renew Subscription screen

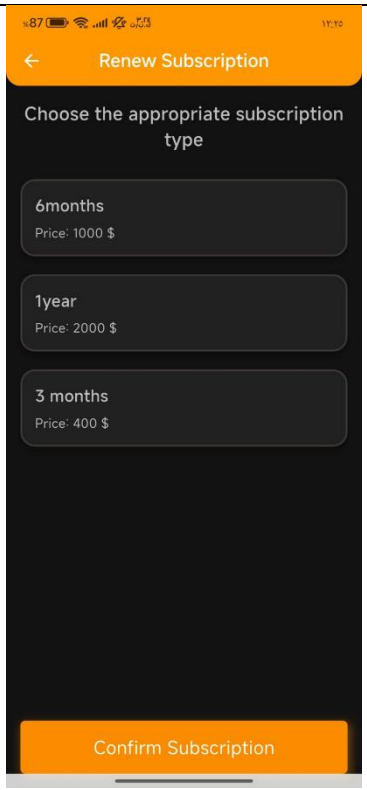


Figure 93 subscription type

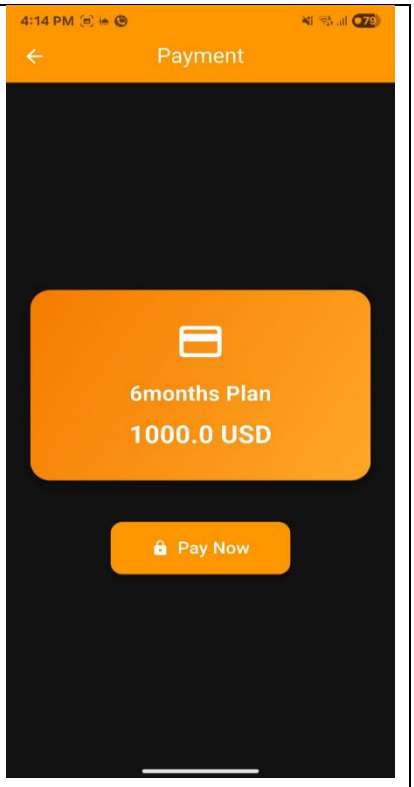


Figure 94 payment

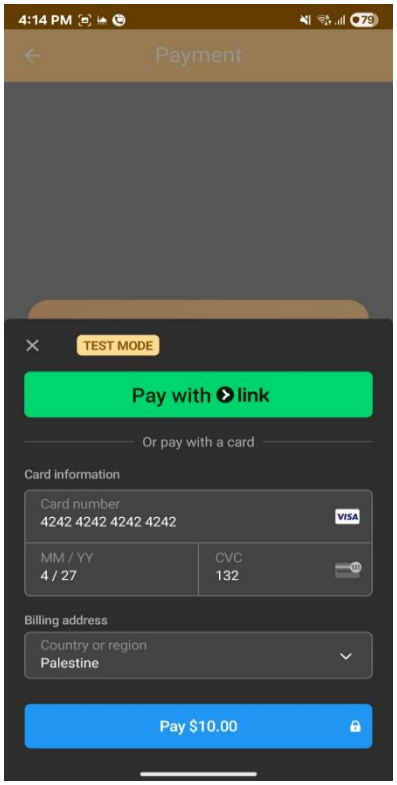


Figure 95 payment form

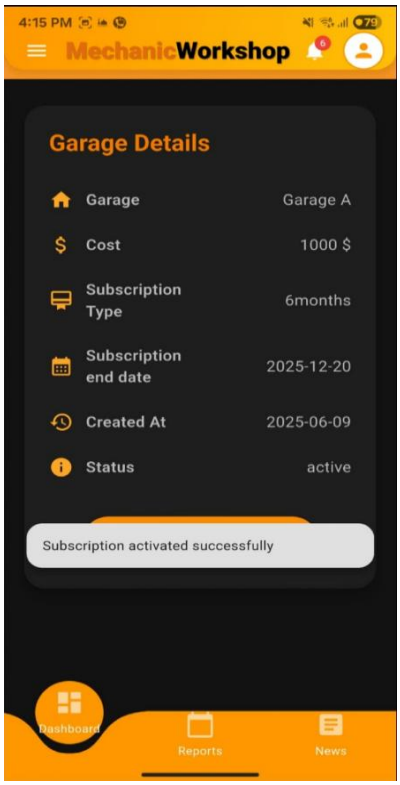


Figure 96 success renew

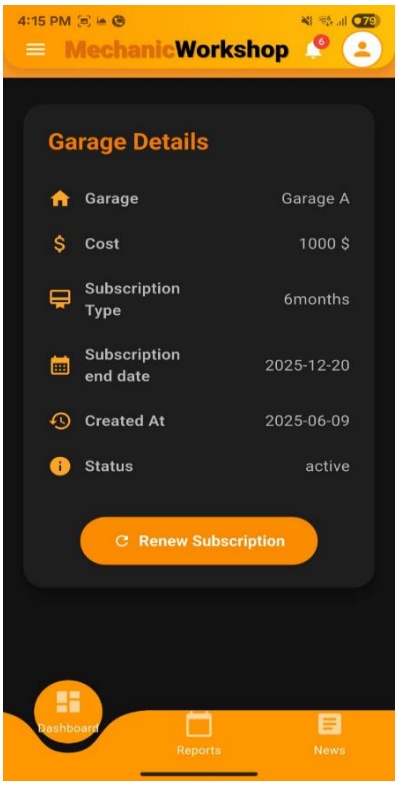


Figure 97 subscription updated

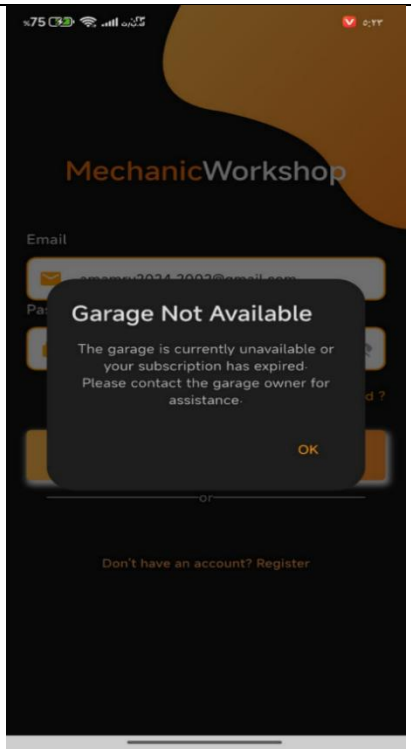


Figure 98 subscription expired

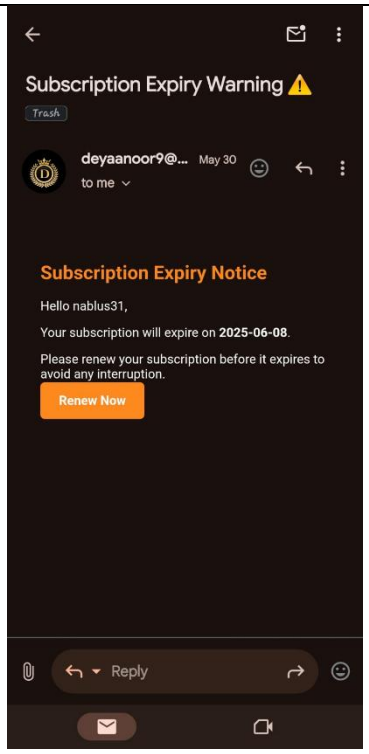


Figure 99 Subscription expiration warning message

5.9.2.8.9 Contact Us Page

Through this page, the garage owner can send messages to the administration in case of any issues or suggestions related to the application.

The page also displays a history of previously sent messages for reference.

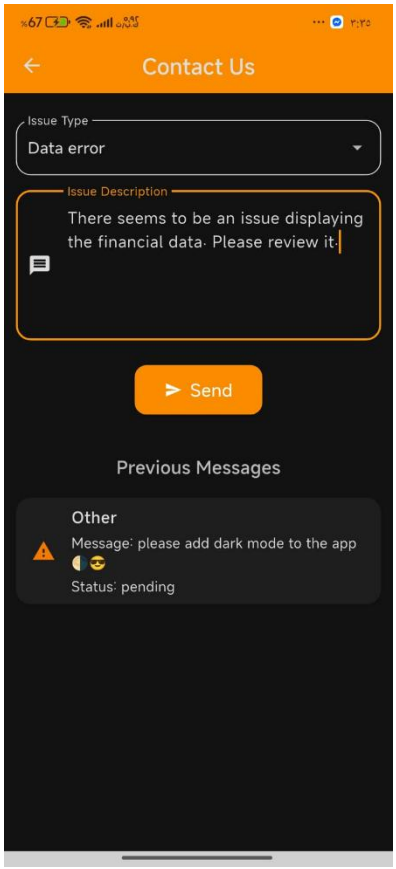


Figure 100 Contact Us page

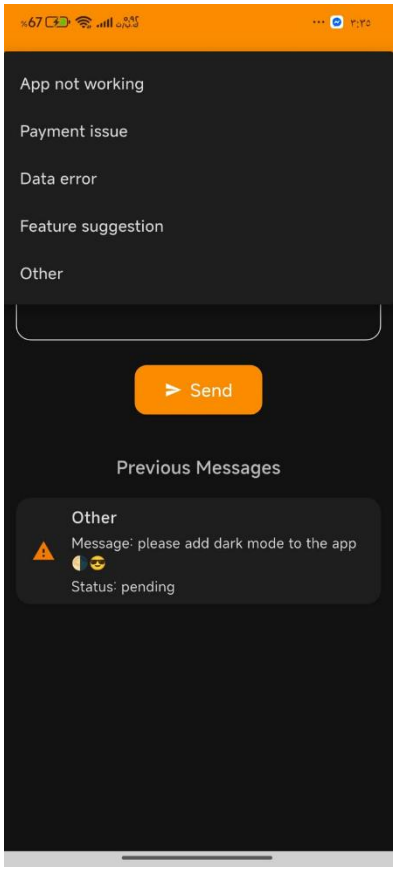


Figure 101 Issue Type

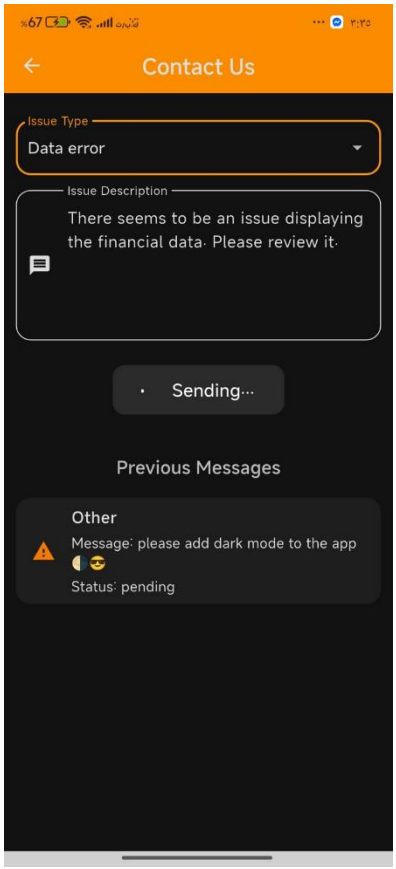


Figure 102 Sending contact us

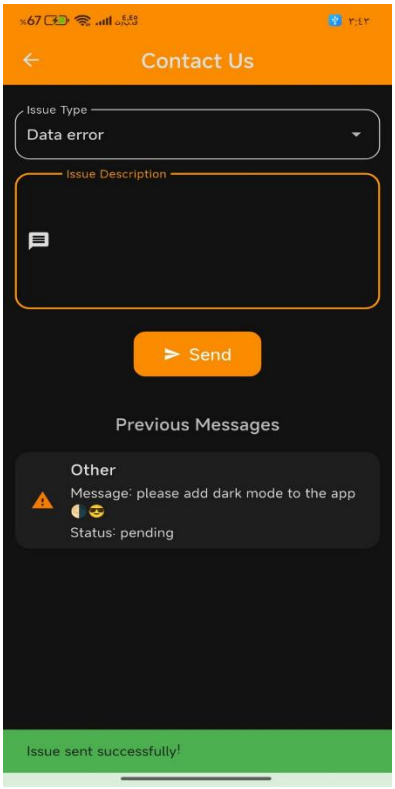


Figure 103 success send problem

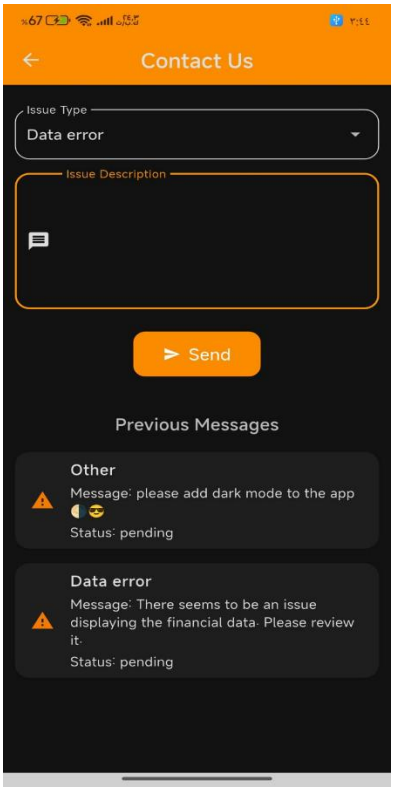


Figure 104 view contact us after sending

5.9.2.8.10 Settings Page

This page includes the following options:

- Account: Manage personal account details.
- Theme Mode: Switch between Dark Mode and Light Mode.
- Change Language: Choose between Arabic and English.

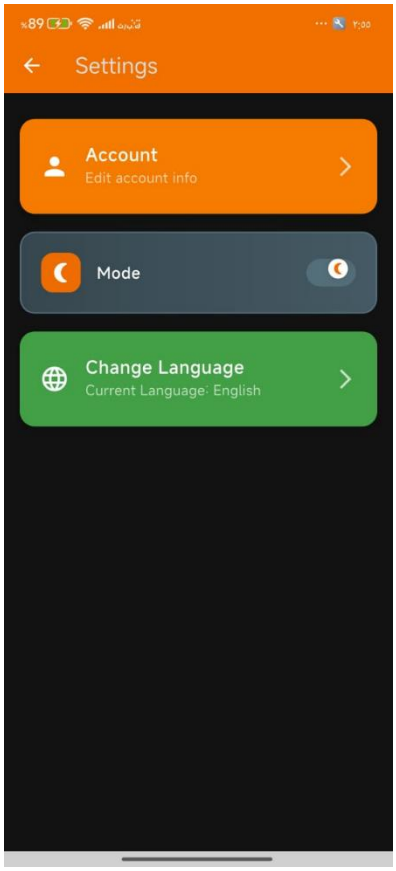


Figure 105 settings English view

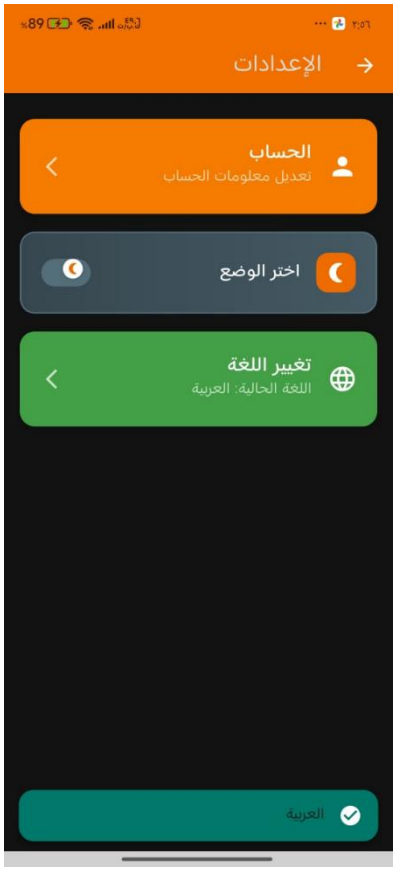


Figure 106 settings Arabic view

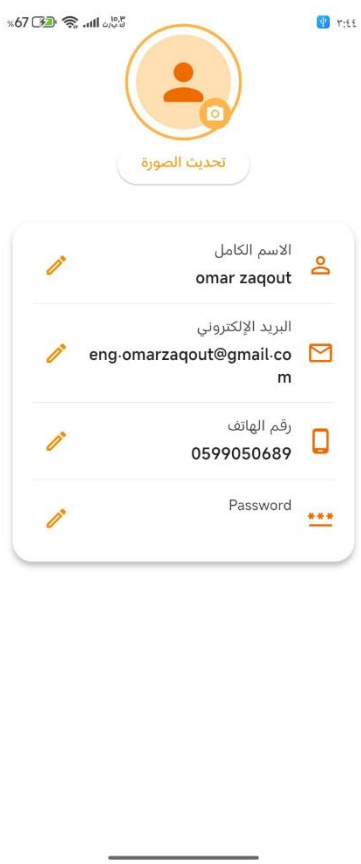


Figure 107 Account Page

5.9.2.8.11 Notifications Page

- This page displays real-time notifications sent to the admin when:
  - A report is submitted.
  - A client sends a help request.
- Features:
  - Swipe to delete: Swiping a notification prompts a confirmation before deletion.
- Tap to view:
  - If the notification is related to a help request, tapping it navigates to the Requests Page and removes the blue “unread” indicator.
  - If the notification is related to a report, tapping it navigates directly to the Report Details Page of the selected report, and the unread indicator is removed.

Figure 108 Notification

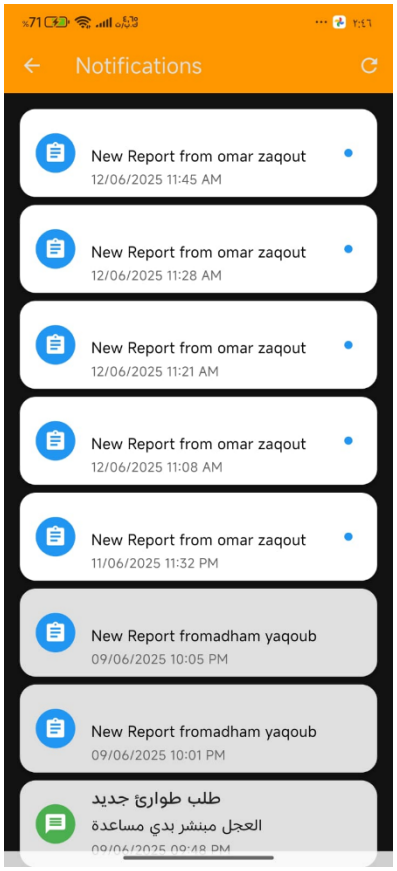


Figure 109 All Notification

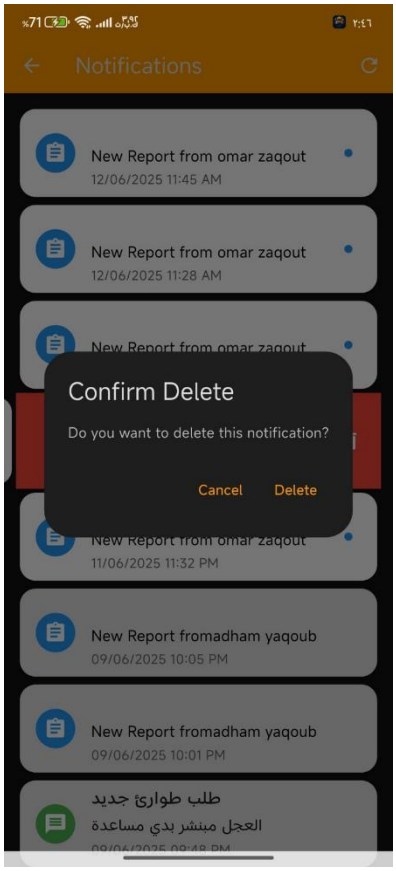


Figure 110 swipe left notification

Figure 111 notification emergency request

5.9.2.9 *Employee Pages*

5.9.2.9.1 Instant News Page

This page displays announcements and real-time news published by the garage owner.

- Each announcement includes:
  - Publish date
  - Title
  - Content
  - Instant notifications are sent to all employees when a new announcement is posted.

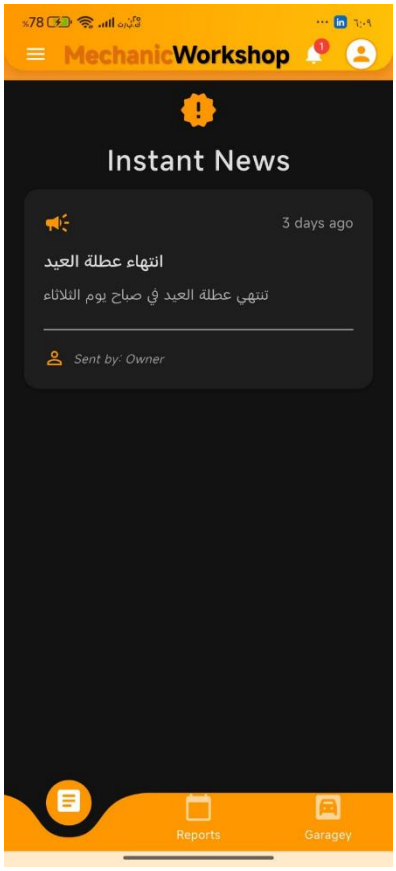


Figure 112 News Page

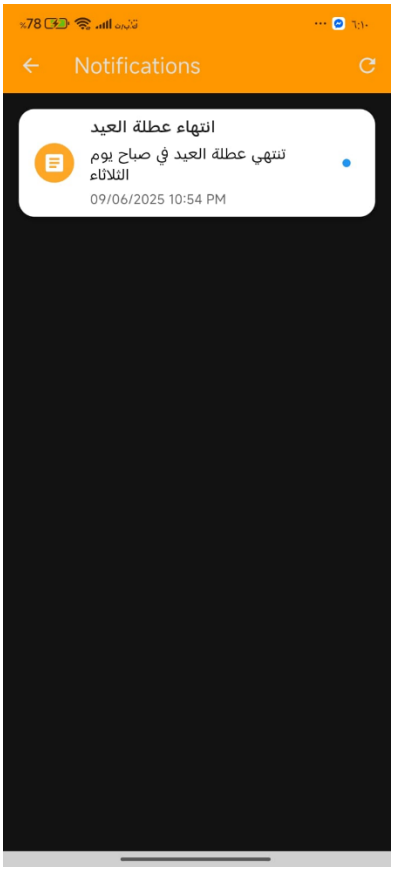


Figure 113 Notification page

5.9.2.9.2 Reports Page

Employees can:

- View all repair reports related to their garage.
- See full report details, including:
  - Car owner’s name
  - License plate number
  - Problem title

- Symptoms
- Used parts Repair
- Description
- Employees cannot delete or edit reports.

- They can add a new report using the same process as the owner:
  - Enter car and problem details.
- Use the AI model (built with Python and TensorFlow) to analyze the issue and generate a solution based on:
  - Car makes, model, year
  - Problem title
  - Car symptoms
  - Optionally use voice input to fill in the repair description for convenience.
  - Upon submission:
    - The report is saved.
    - A success or failure popup is shown.
    - The garage owner receives a notification.

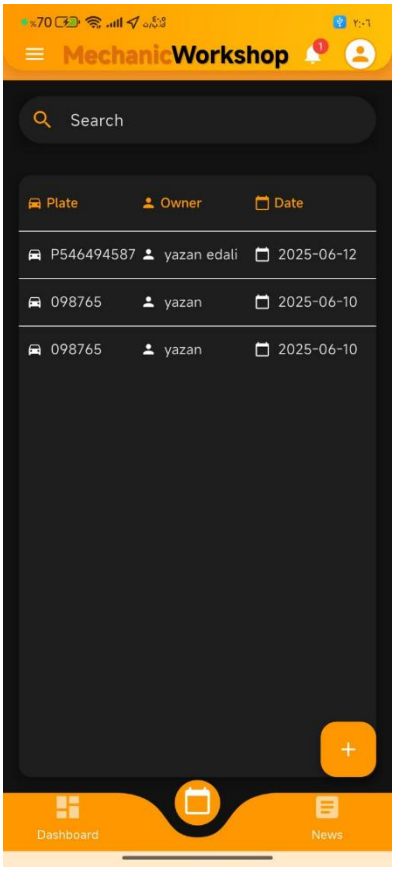


Figure 114 Report screen

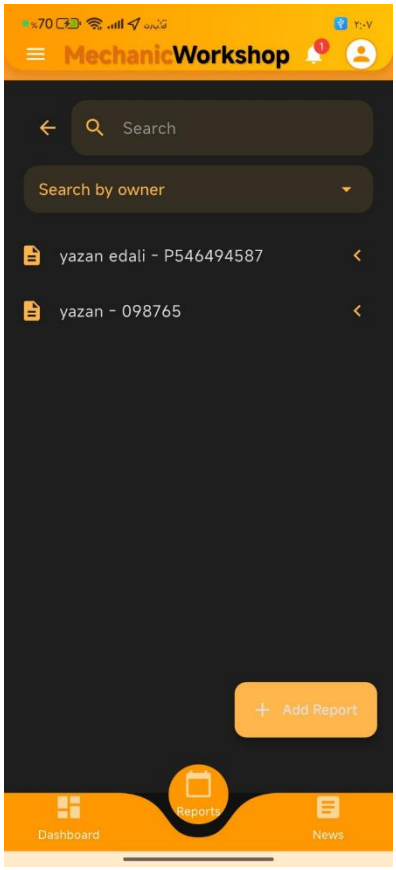


Figure 115 exist client report

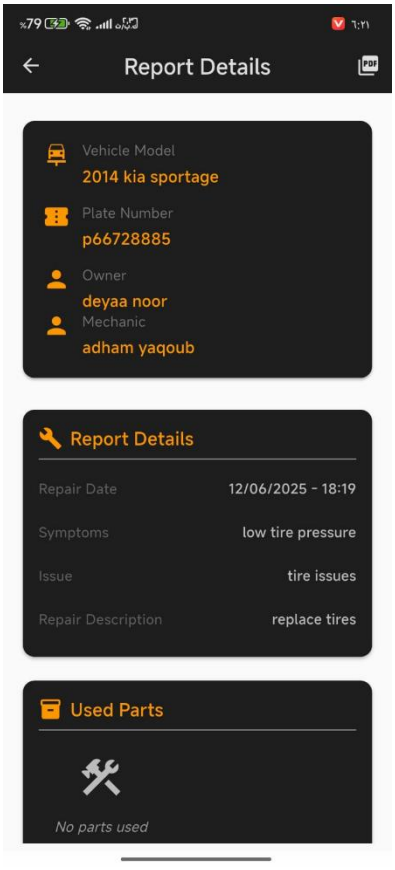


Figure 116 report details

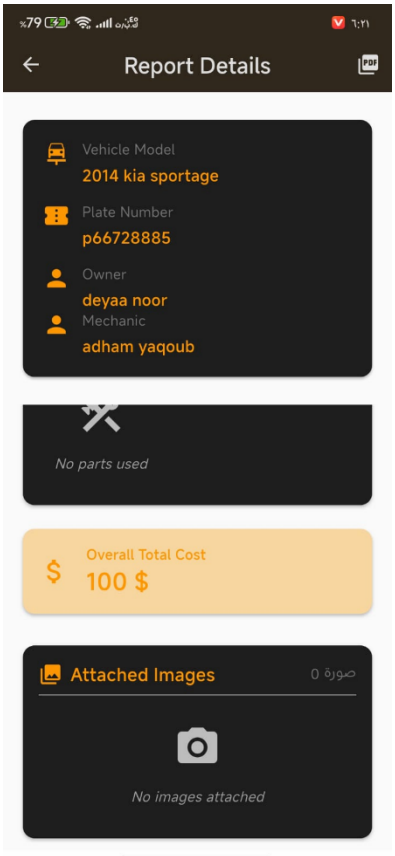


Figure 117 con. report details

5.9.2.9.3 Garage Page

- This page displays detailed information about the garage, including:
  - Garage Name
  - Owner's Name
  - Owner's Email: Tapping it opens the Gmail app.
  - Owner's Phone Number: Tapping it opens the phone dialer.
  - Total Number of Completed Repairs
  - Owner's Total Salary

The page provides quick access to contact the garage owner and view basic operational stats related to the garage.

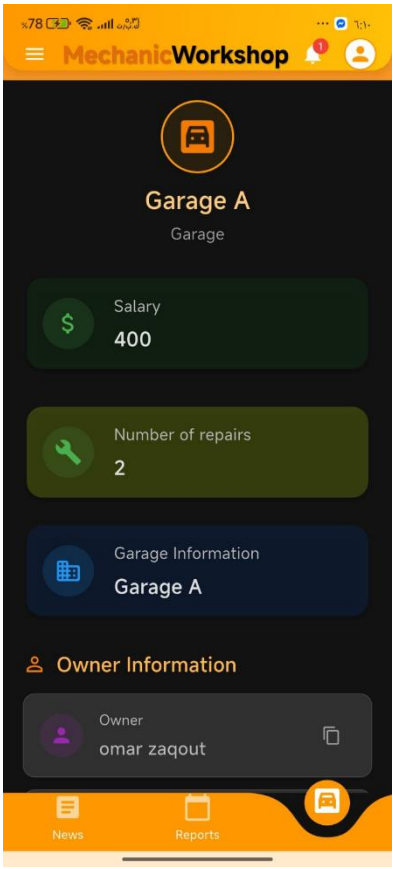


Figure 118 garage page

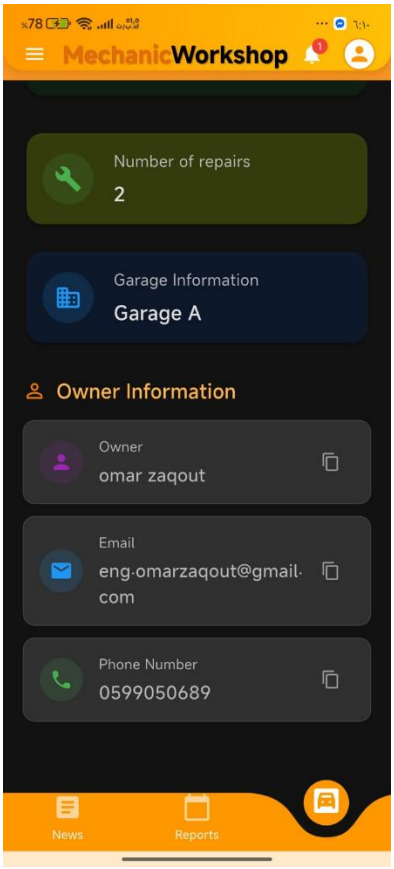


Figure 119 cont. garage screen

5.9.2.9.4 Settings Page

This page allows the user to manage their personal and app preferences:

- Personal Information: View and update personal details such as full name, email, and phone number.
- Theme Mode: Toggle between Dark Mode and Light Mode.
- Change Language: Switch the application's language between Arabic and English.
- Contact Info: Navigate to the page that shows garage contact details.
- Garage Name, Owner Name, Email, and Phone Number
- Interactive Elements:
    - Tapping the phone number opens the phone dialer.
    - Tapping the message icon opens WhatsApp or the SMS app.
    - Tapping the email opens the default email application (e.g., Gmail).
    - Tapping the location/map opens the garage's location on Google Maps.

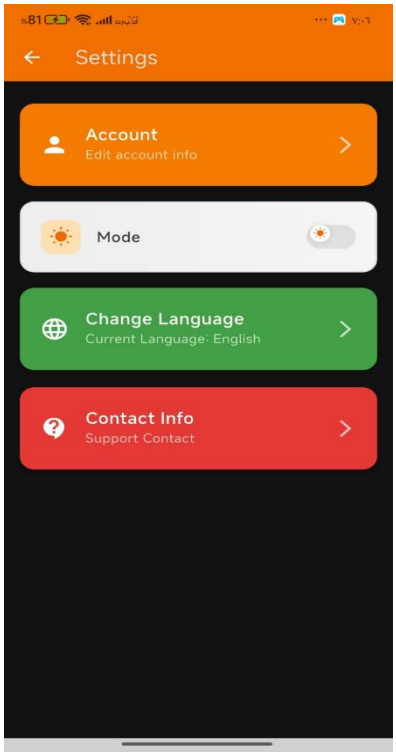


Figure 120 settings

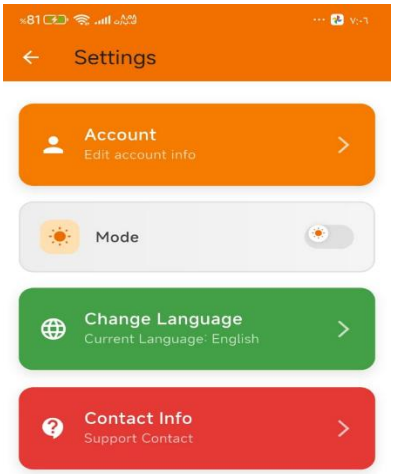


Figure 121 light theme

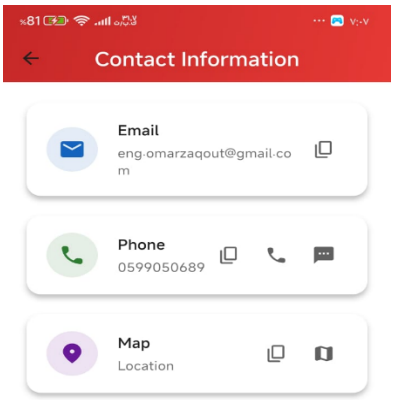


Figure 122 contact information owner

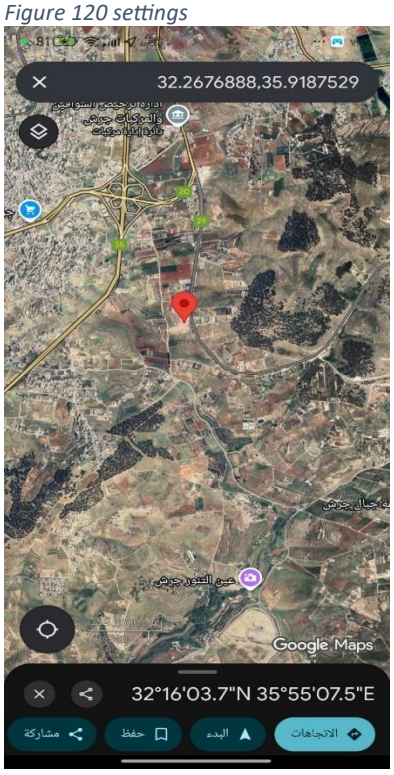


Figure 123 location garage

Figure 124 click phone

5.9.2.10 *Client Pages*

5.9.2.10.1 Garage Page

- Displays all the garages the client is registered with.
- When clicking on any garage, it navigates to a page with two tabs:
    - Service History:
        - Shows all repairs done by that garage.
        - Clicking on any repair shows the detailed report of that repair.
    - Requests:
        - Shows all the help requests the client has made.
        - Clicking on any request displays:
            - The garage location
            - The client's current location
            - The chat conversation between the client and the garage regarding the request

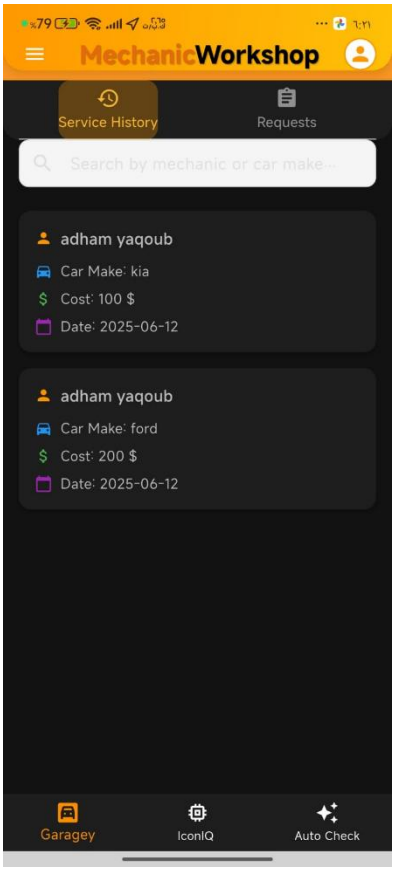


Figure 125 service history

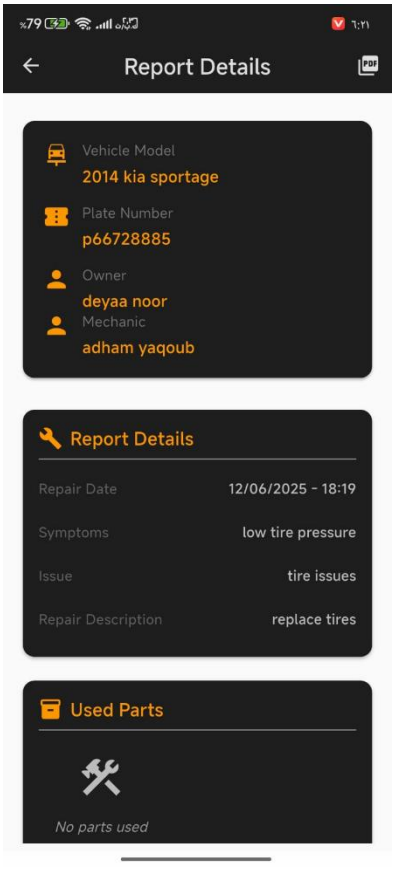


Figure 126report details

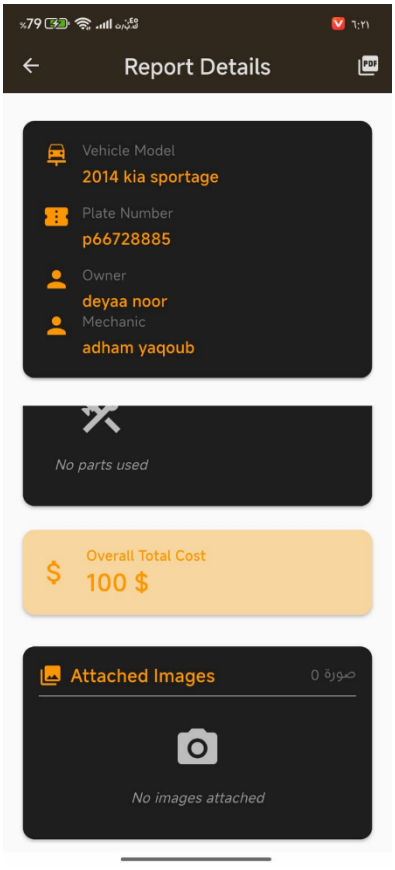


Figure 127 cont. report details

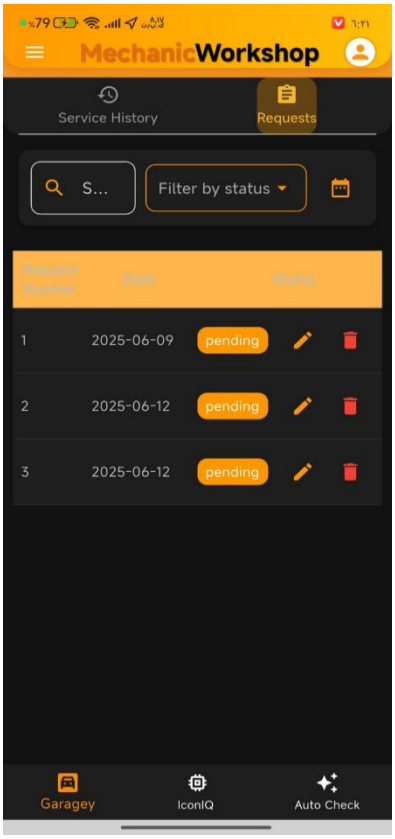


Figure 128 my emergency request

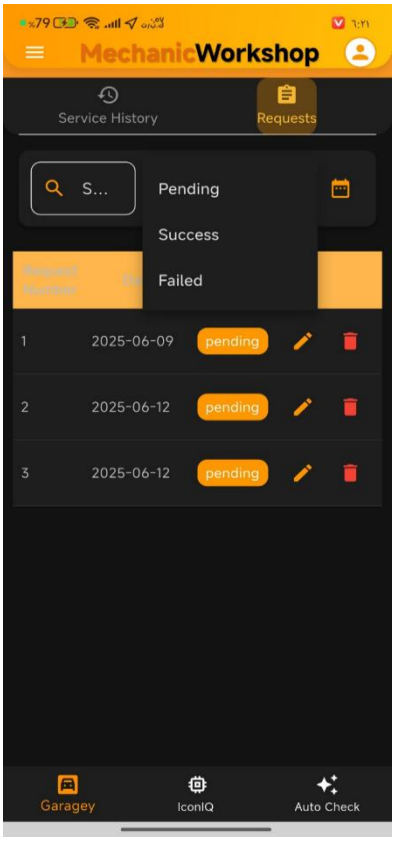


Figure 129 filter status

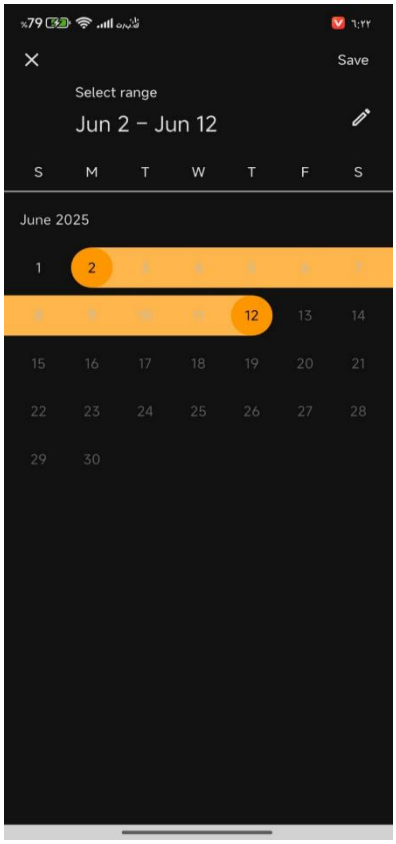


Figure 130 filter date

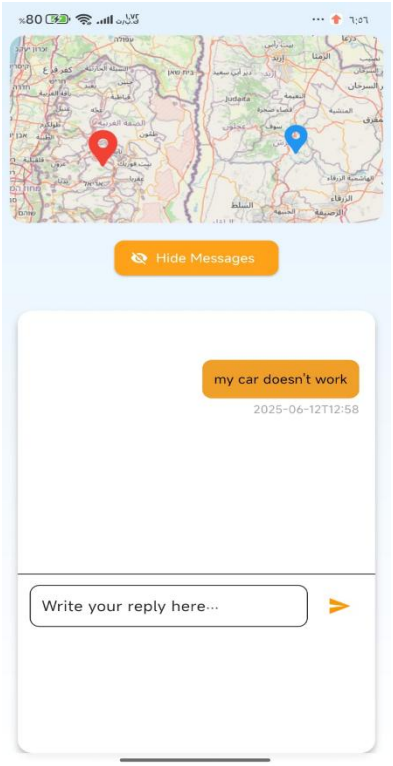


Figure 131 conversation request

Figure 132 location garage and client need emergency

5.9.2.10.2 Icon IQ Page

This page allows the user to upload or capture an image of the car dashboard symbols.

- The image is processed using a Roboflow model to detect and identify the symbols.
- Each detected symbol is displayed with its name.
- The user can click on any symbol to see more detailed information about it.

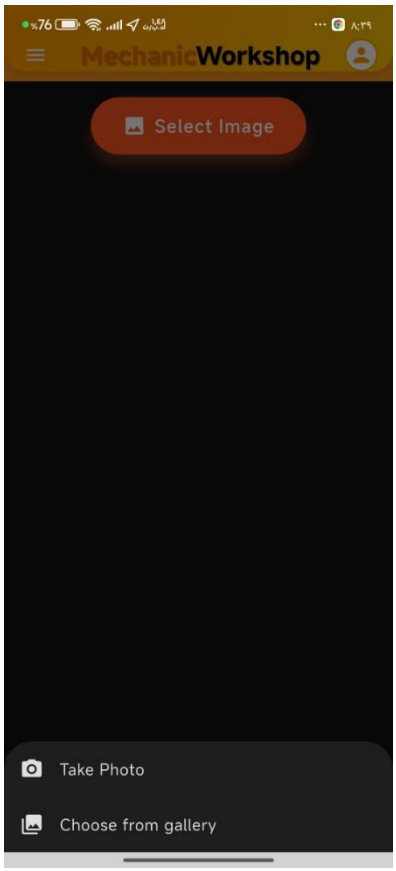


Figure 133 upload image to process icon

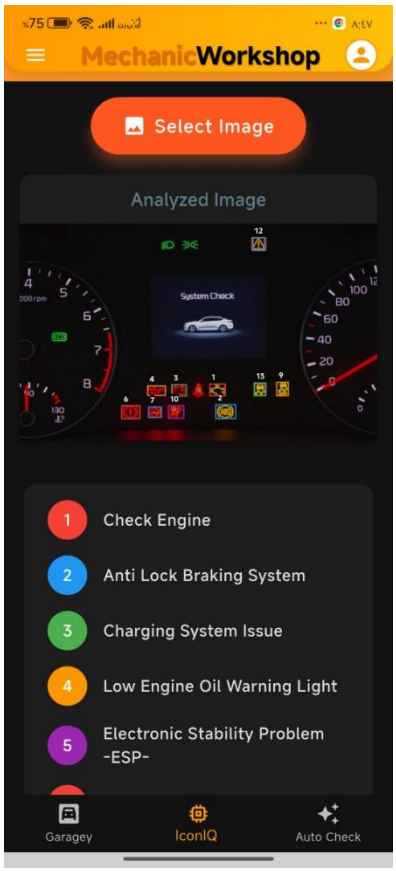


Figure 134 the image process icons

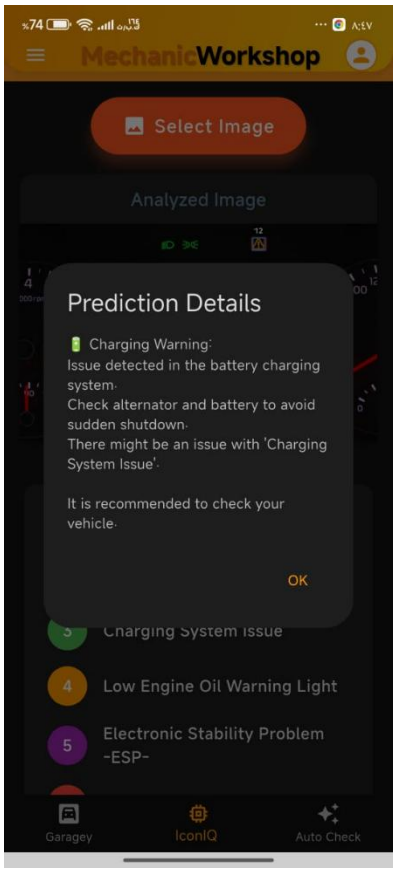


Figure 135 prediction details

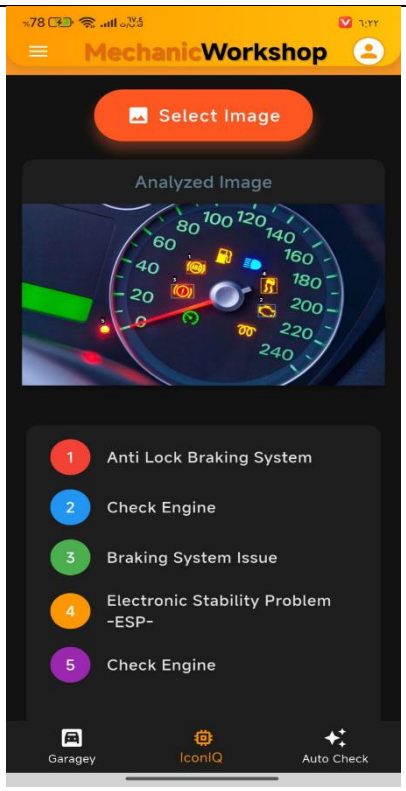


Figure 136image process icons2

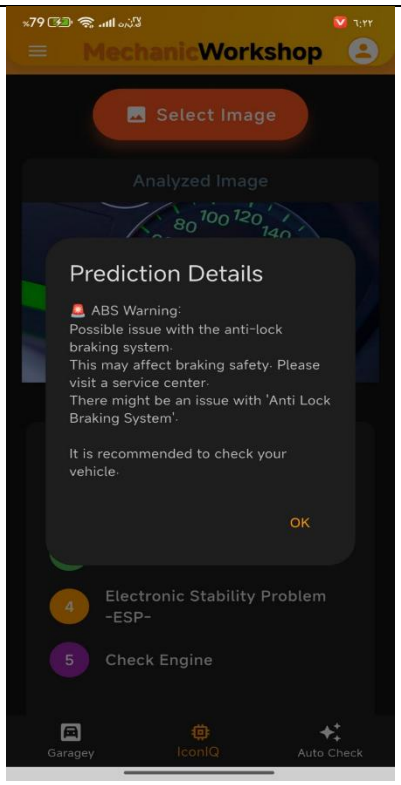


Figure 137 prediction details 2

5.9.2.10.3 Auto Check Page

- We trained an AI model using Python and TensorFlow on a dataset of 10,000 records.
- The model analyzes the following inputs:
  - Car Make
  - Car Model
  - Problem Title
  - Symptoms
  - Year
- Based on these inputs, the model provides a suggested solution to the car issue.

Figure 138model to check my car

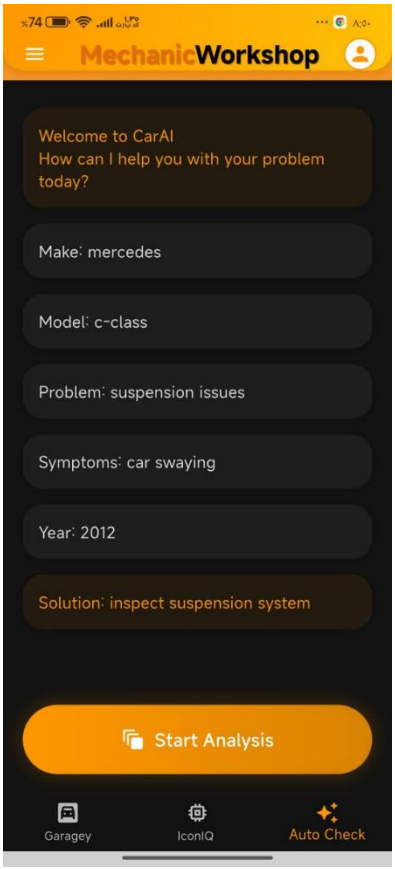


Figure 139 example analysis1

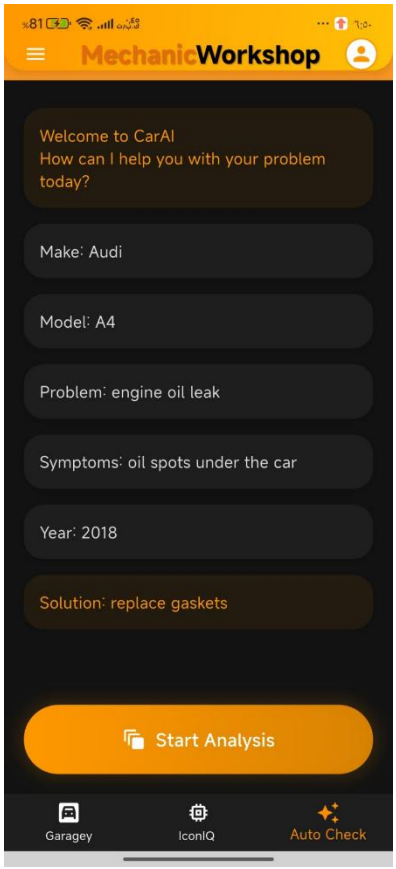


Figure 140 example analysis 2

5.9.2.10.4 Emergency Request Page

- Displays all garages on the map, both subscribed and non-subscribed, along with the user's current location.
- Shows a list of garage names and their distances from the user, sorted in ascending order by distance.
- Garages with active subscriptions are marked with a green checkmark next to their name.
- The user can select any garage to send an emergency request by entering a message.
- When a garage is selected, its marker on the map changes to red to highlight the chosen garage.

Figure 141 select garage A

Figure 142 select garage B

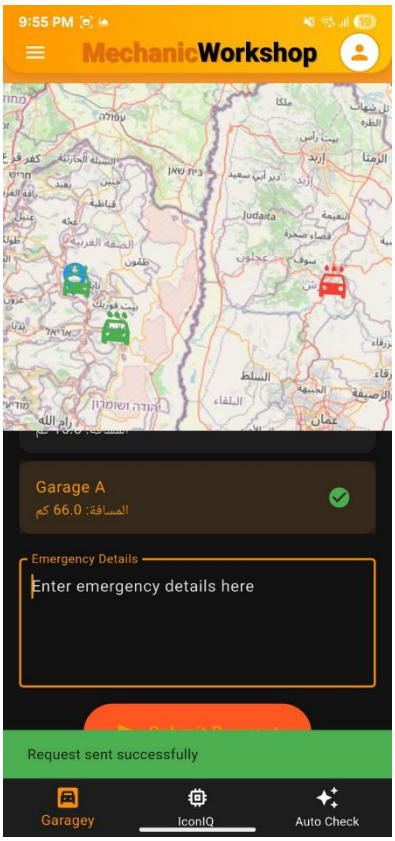


Figure 143 success send request

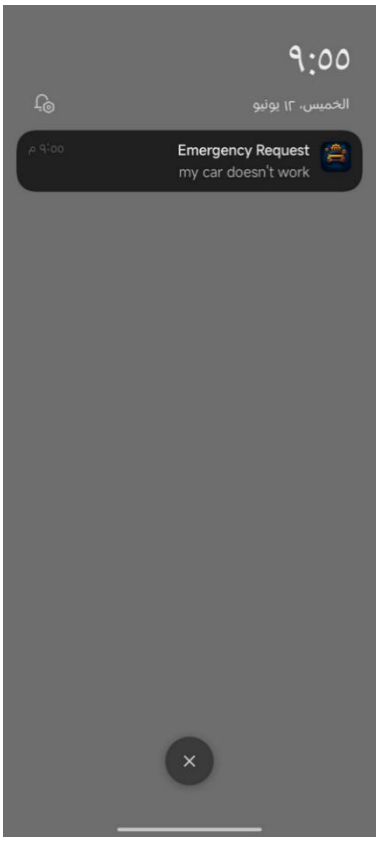


Figure 144 notification emergency request

5.9.3 Contact Us Page

Through this page, the garage owner can send messages to the administration in case of any issues or suggestions related to the application.

The page also displays a history of previously sent messages for reference.

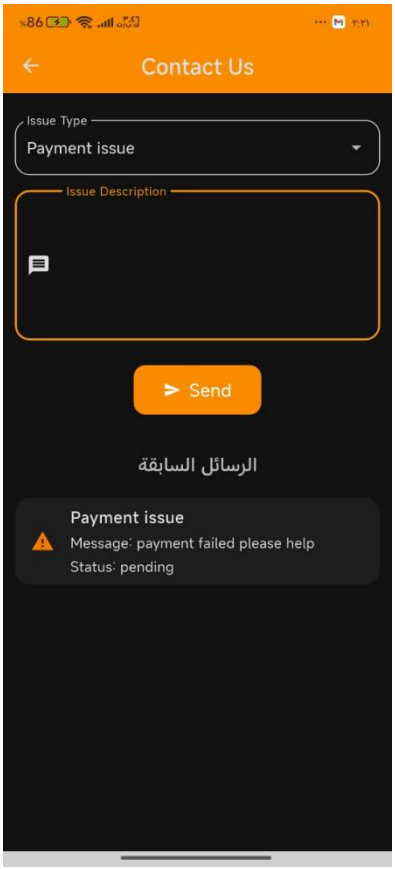


Figure 145 contact Us Page

5.9.4 Settings Page

This page allows the user to manage their personal and app preferences:

- Personal Information: View and update personal details such as full name, email, and phone number.
- Theme Mode: Toggle between Dark Mode and Light Mode.
- Change Language: Switch the application’s language between Arabic and English.

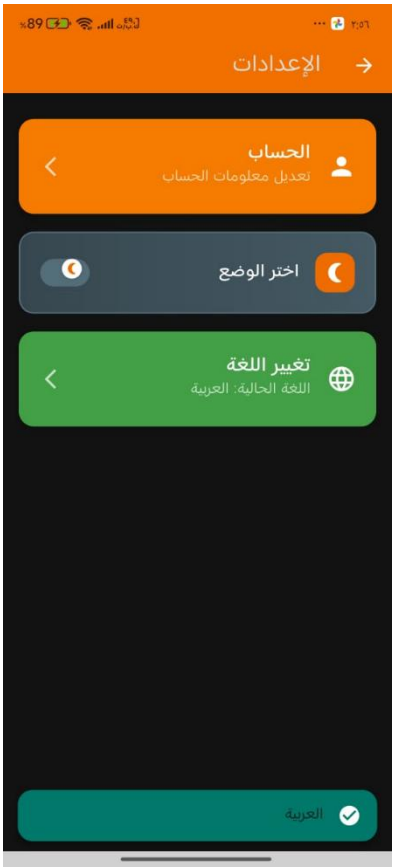


Figure 146 settings Arabic view

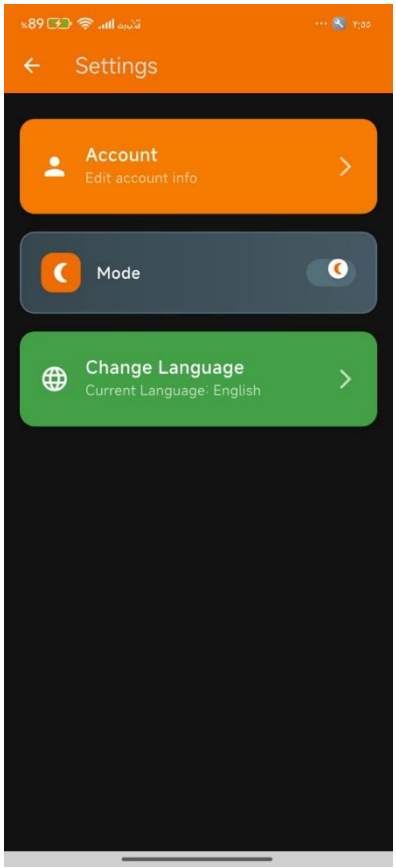
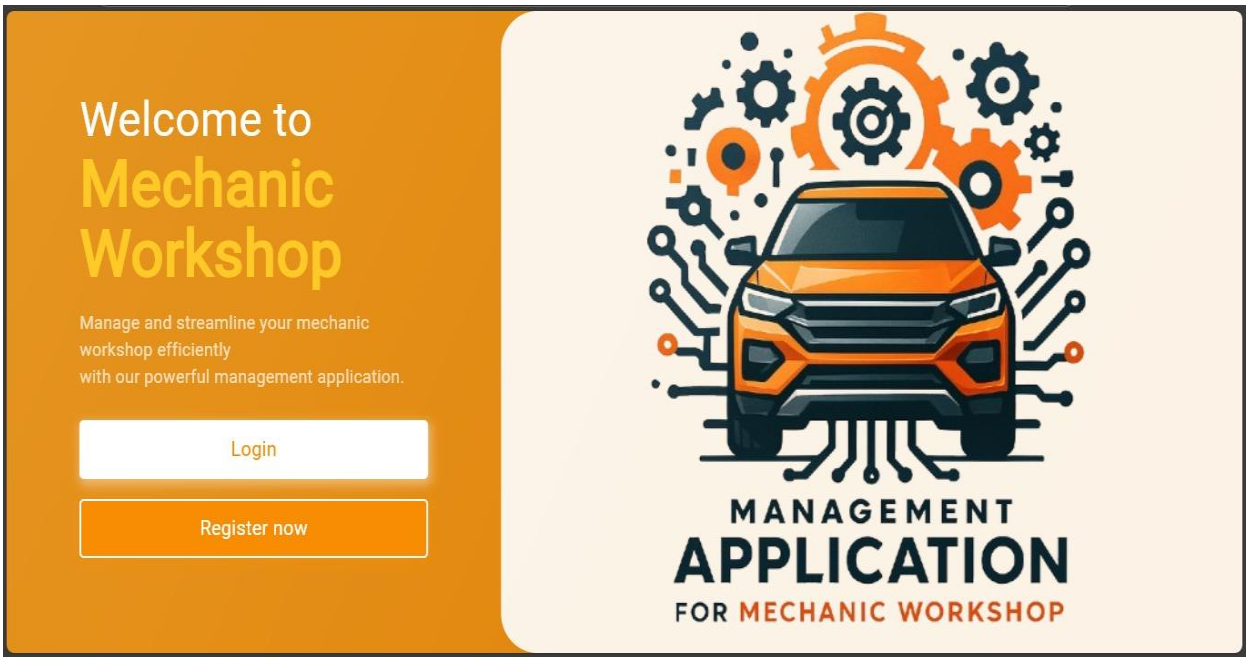


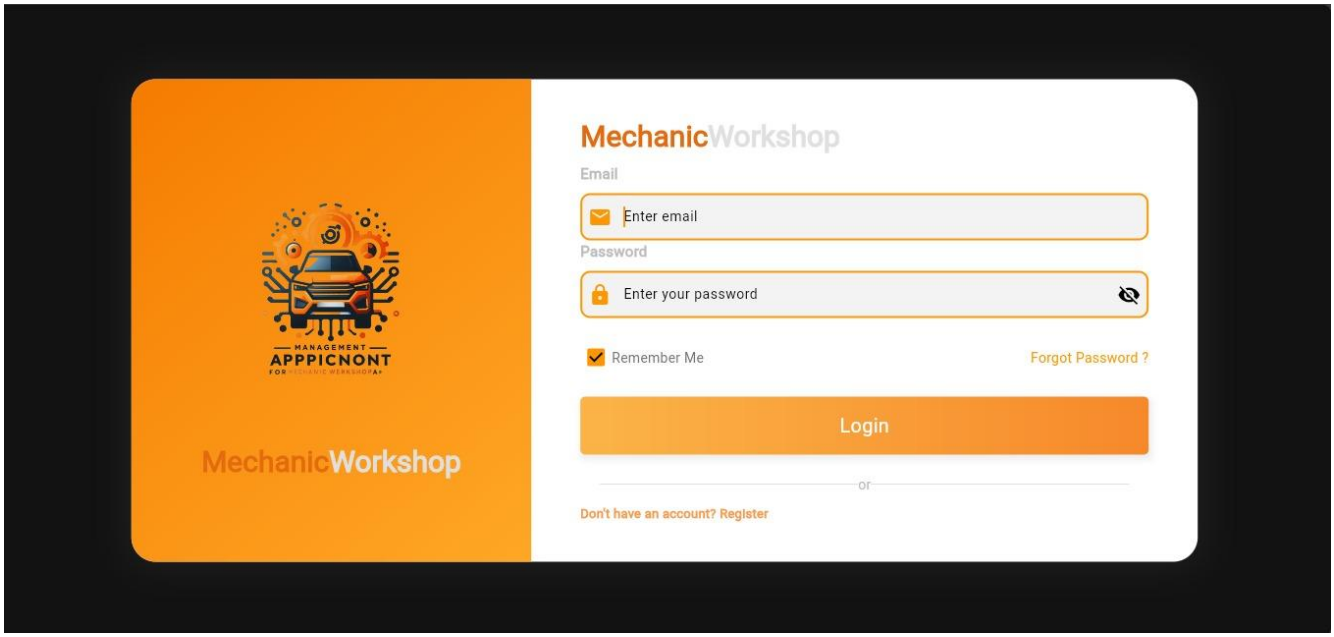
Figure 147 settings English view

5.9.5 Website Pages

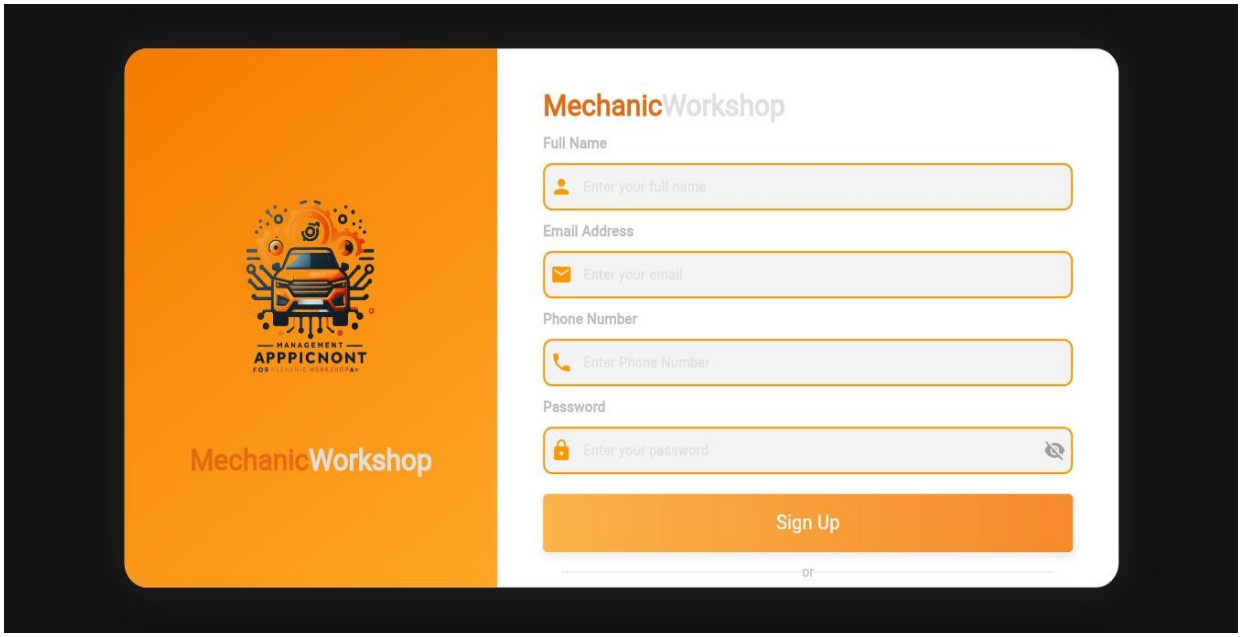
5.9.5.1 Welcome Page



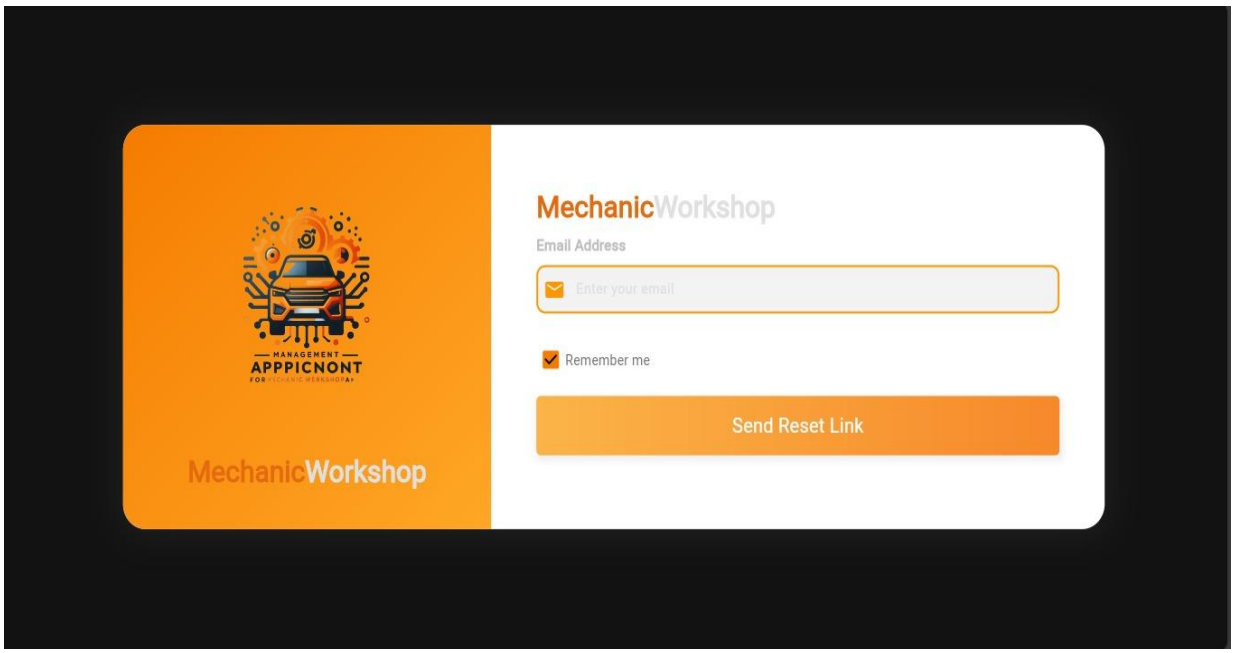
5.9.5.2 *Login Page:*



5.9.5.3 *Register*

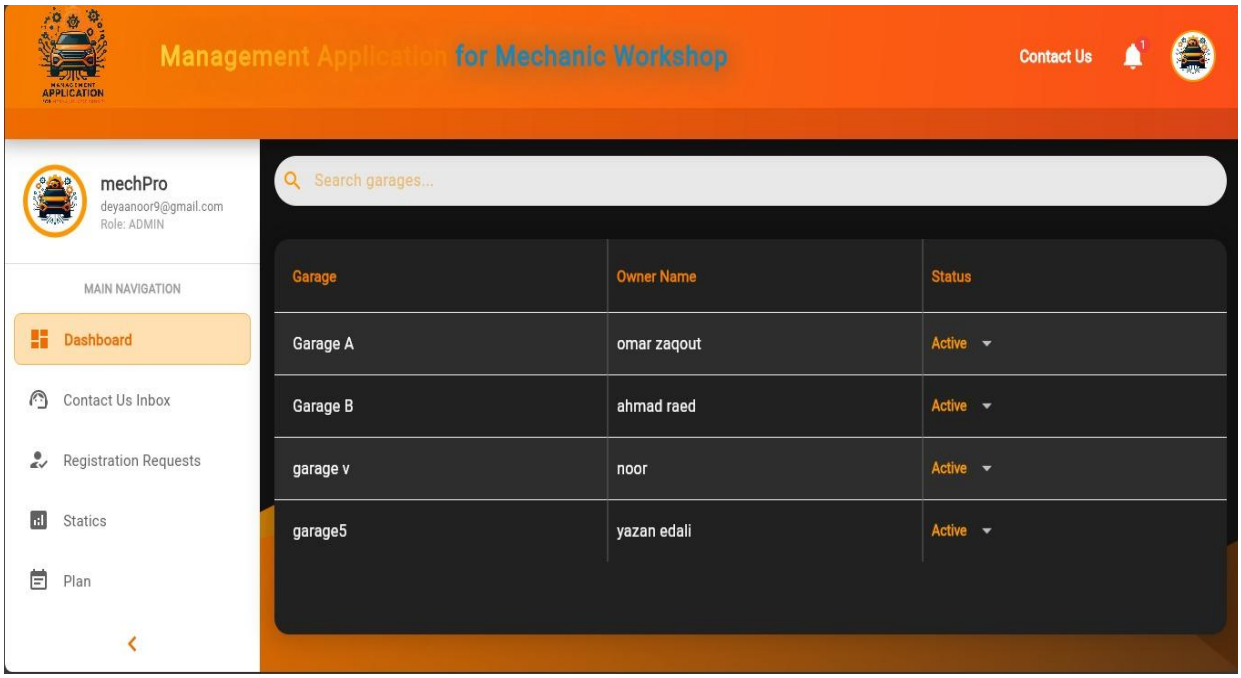


5.9.5.4 *Reset Password*

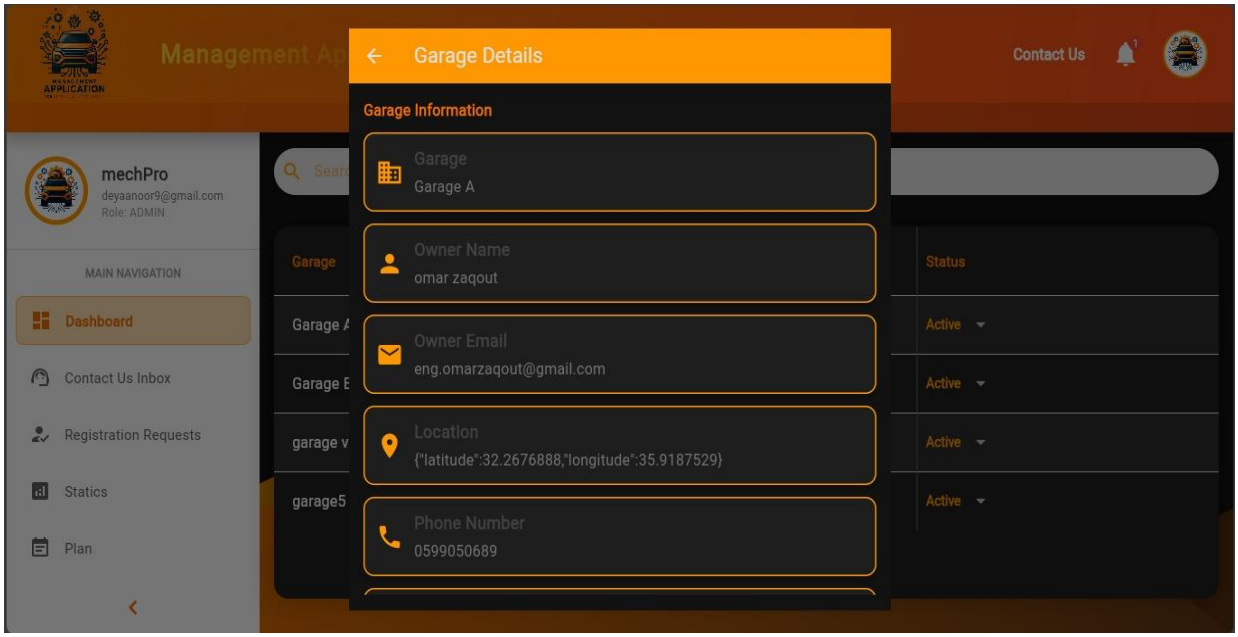


5.9.5.5 *Admin Pages*

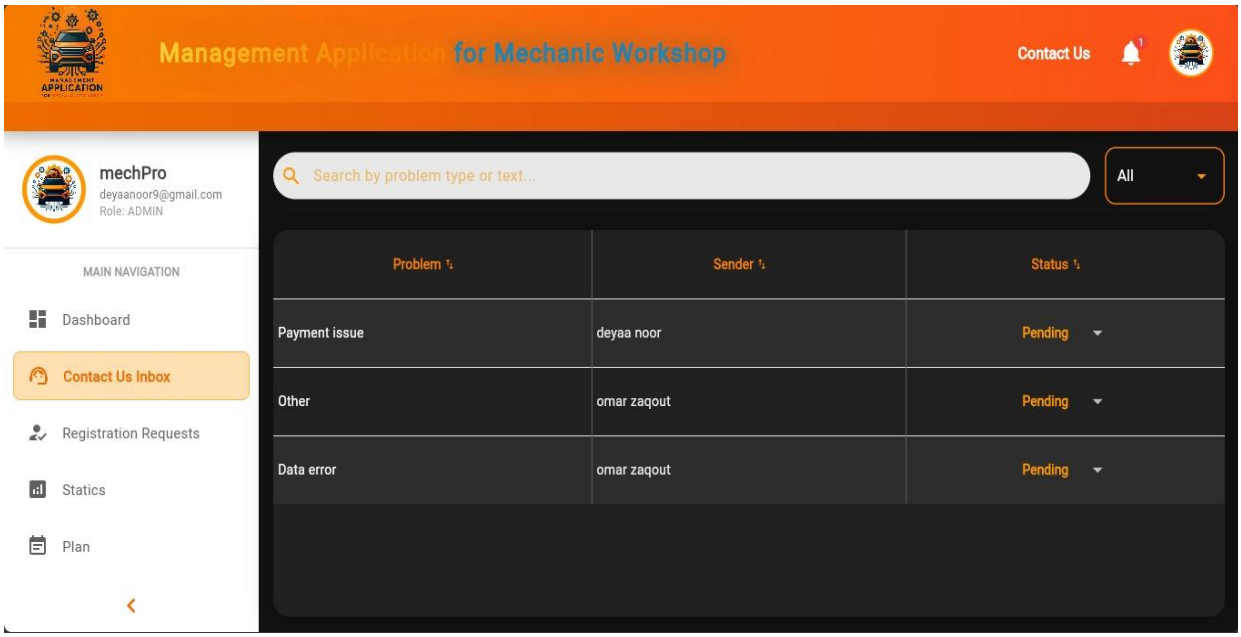
5.9.5.5.1 Dashboard Page



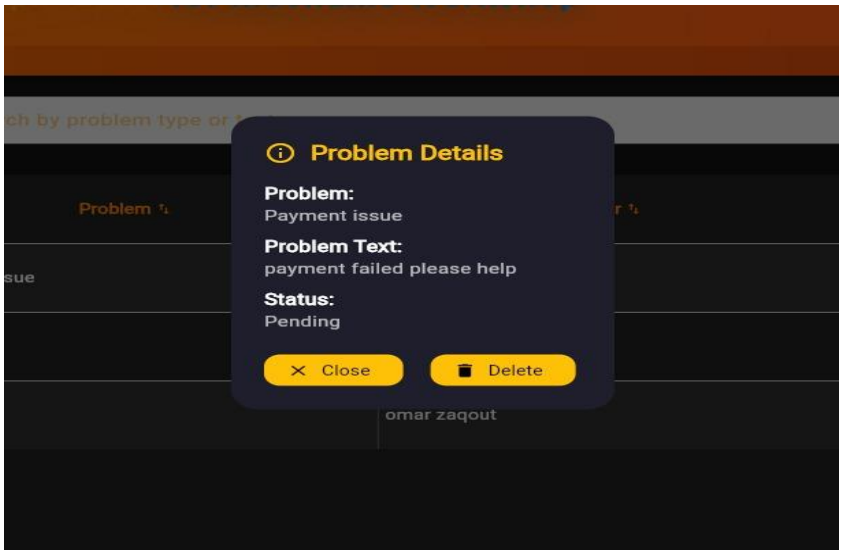
5.9.5.5.1.1 Details



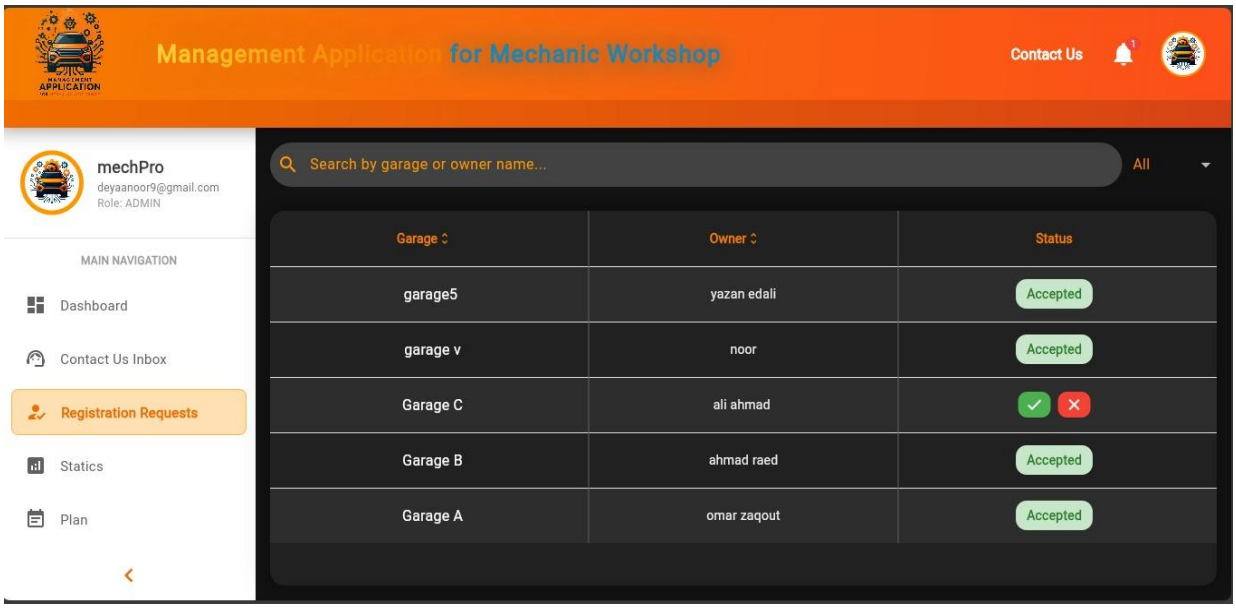
5.9.5.6 *Contact Us Inbox*



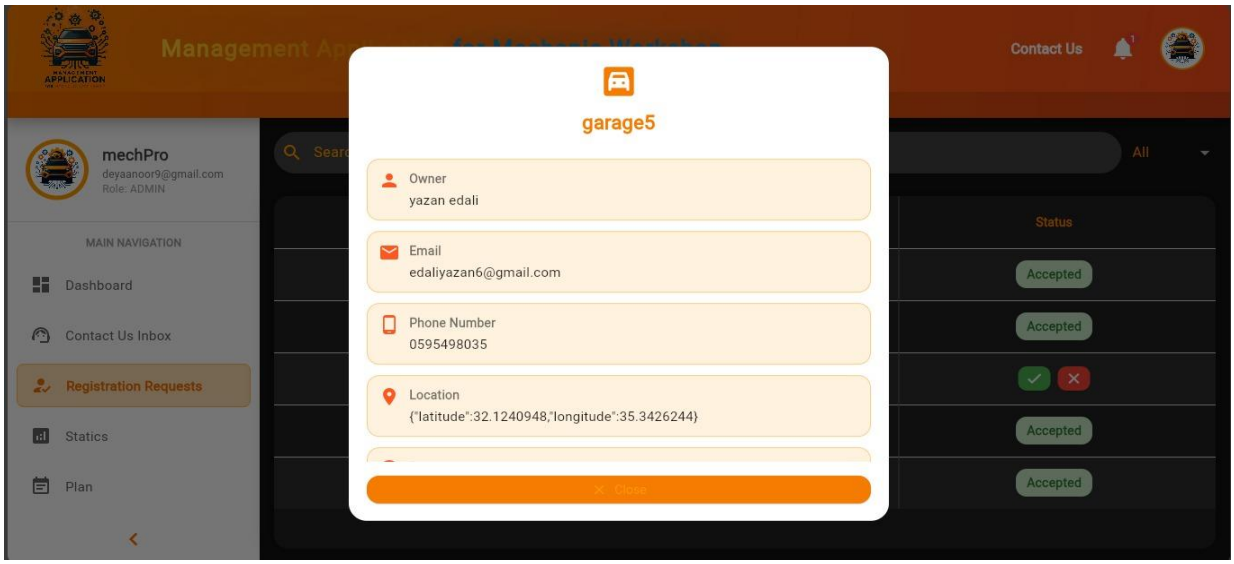
- Problem Details



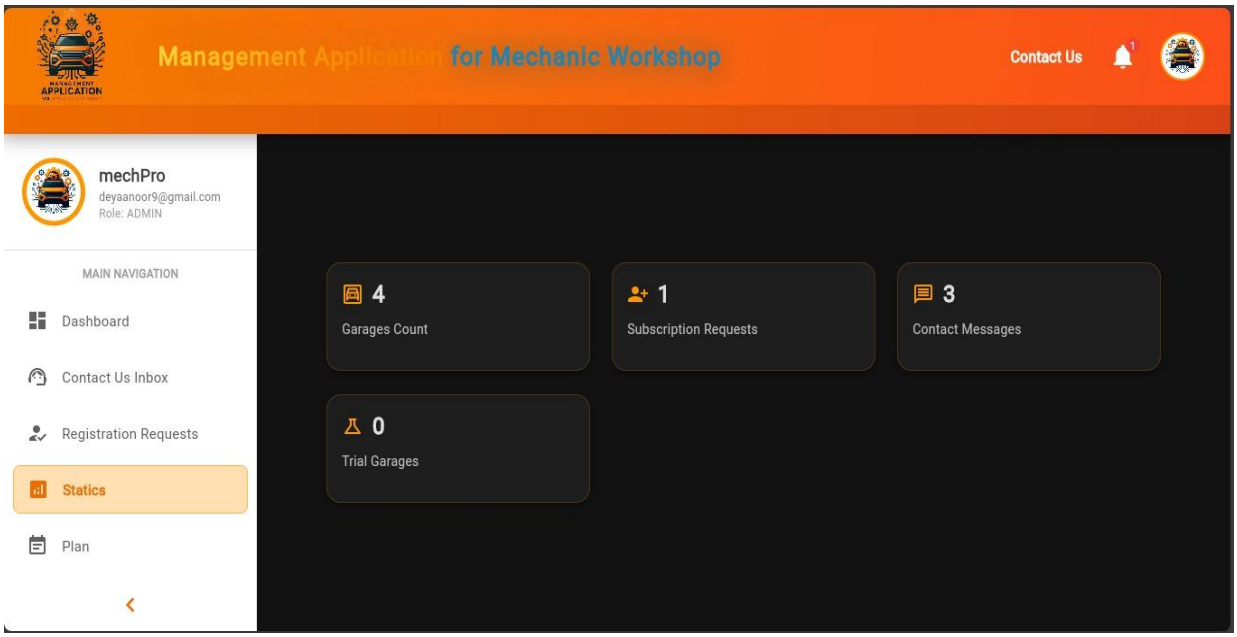
5.9.5.7 *Register Request*



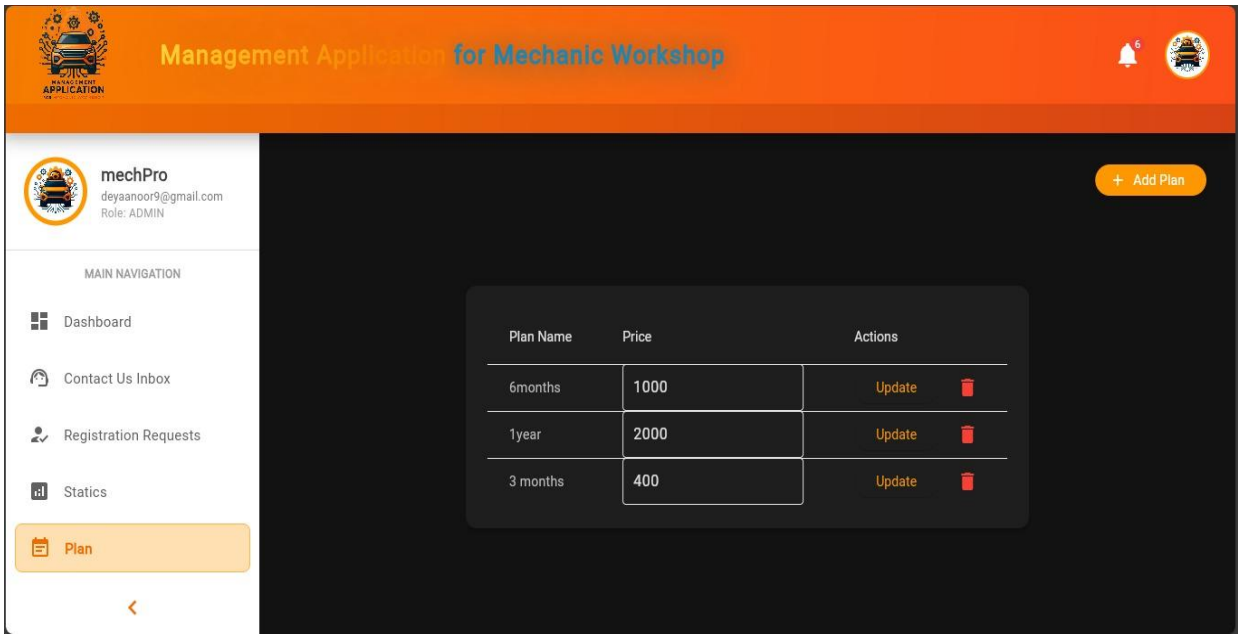
- Register Details



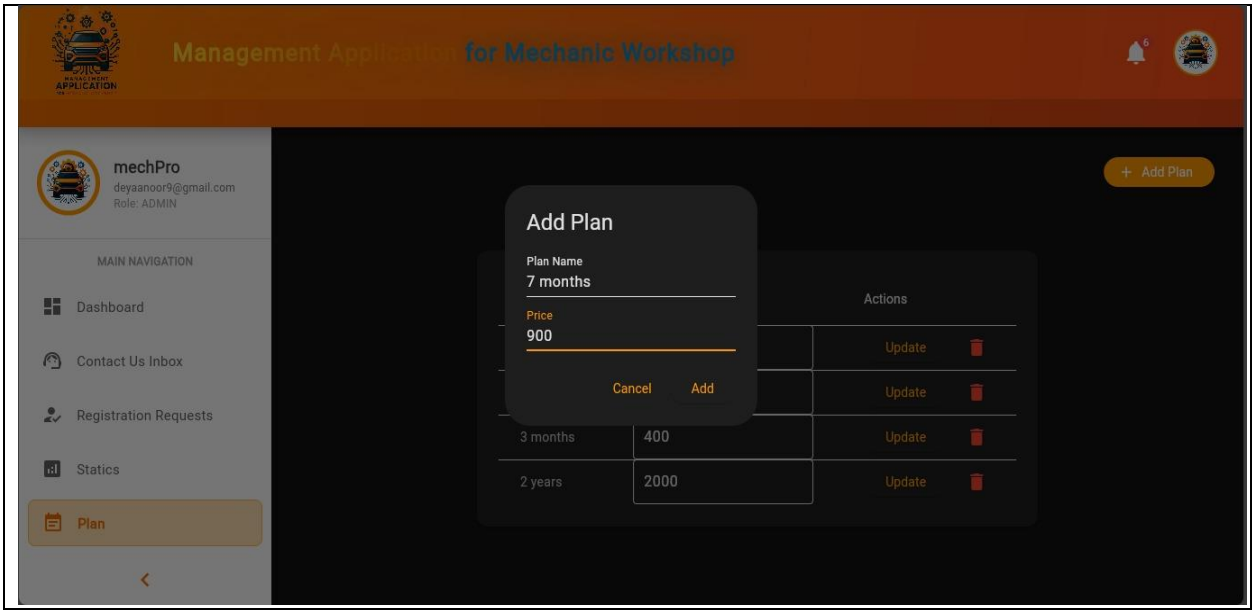
5.9.5.8 *Statics*



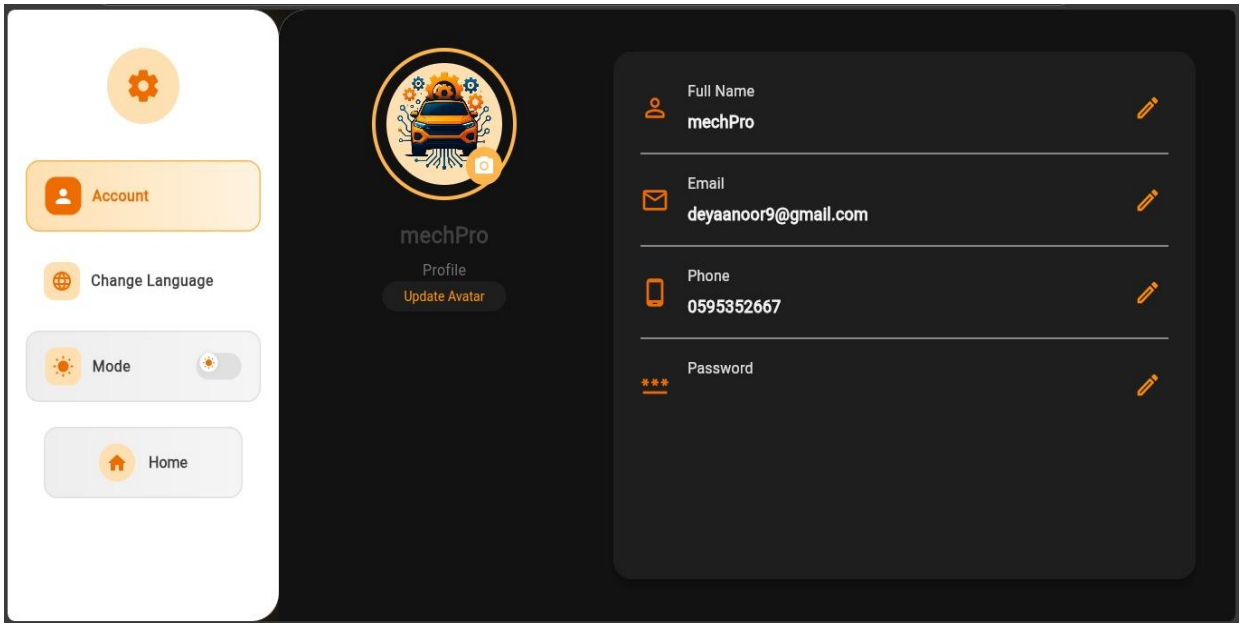
5.9.5.9 *Plan*



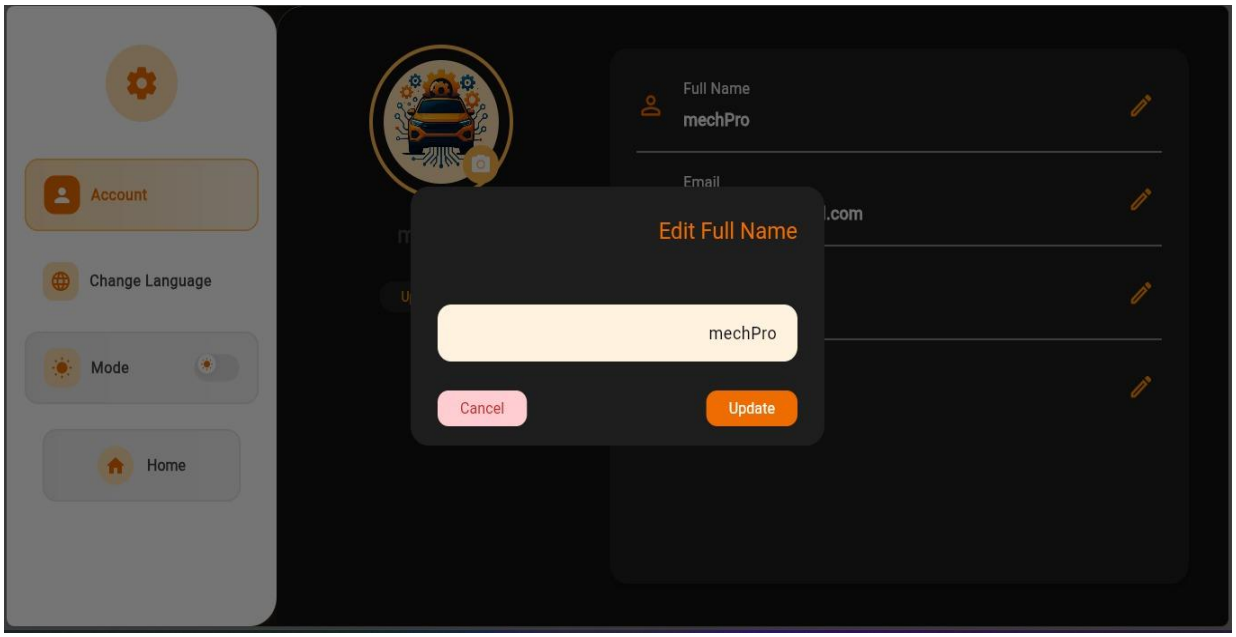
- Add plan



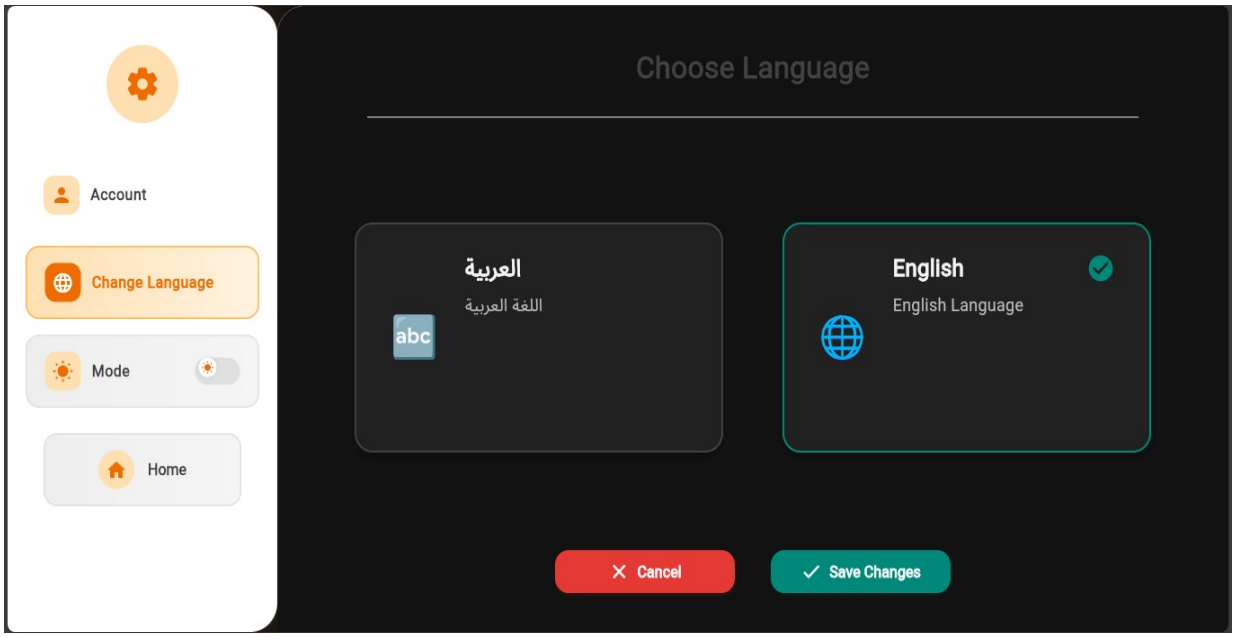
5.9.5.10 *Settings*



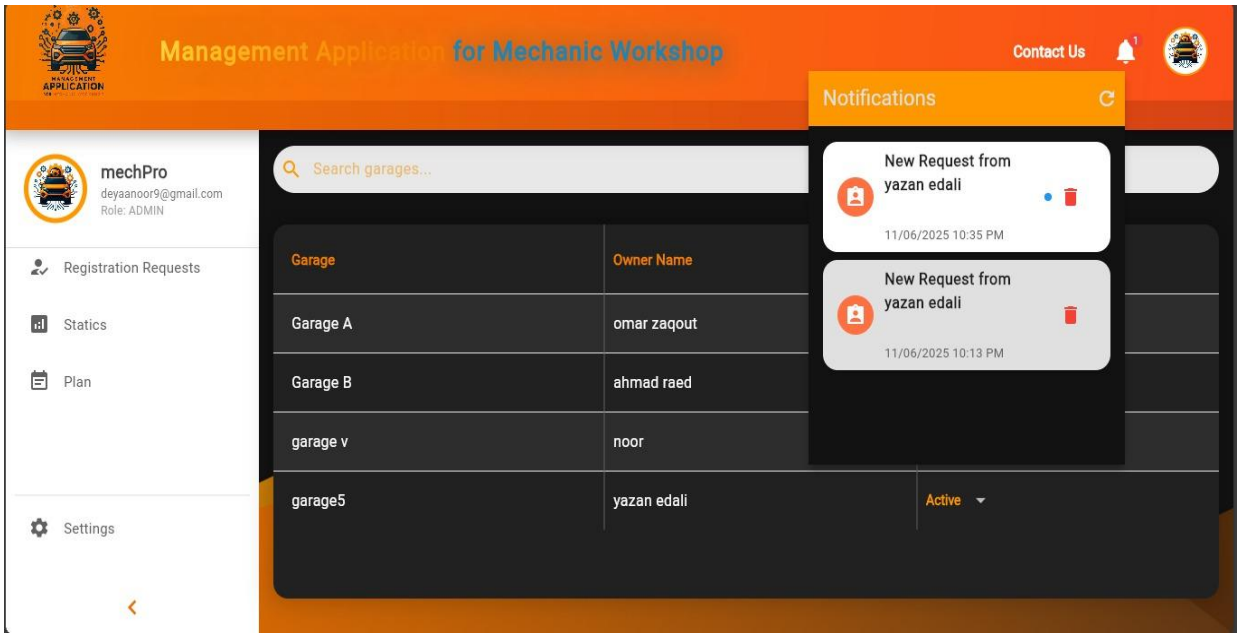
- Edit Full Name


Account
Change Language
Mode
Home
Full Name
mechPro
Email
.com
Edit Full Name
mechPro
Cancel
Update

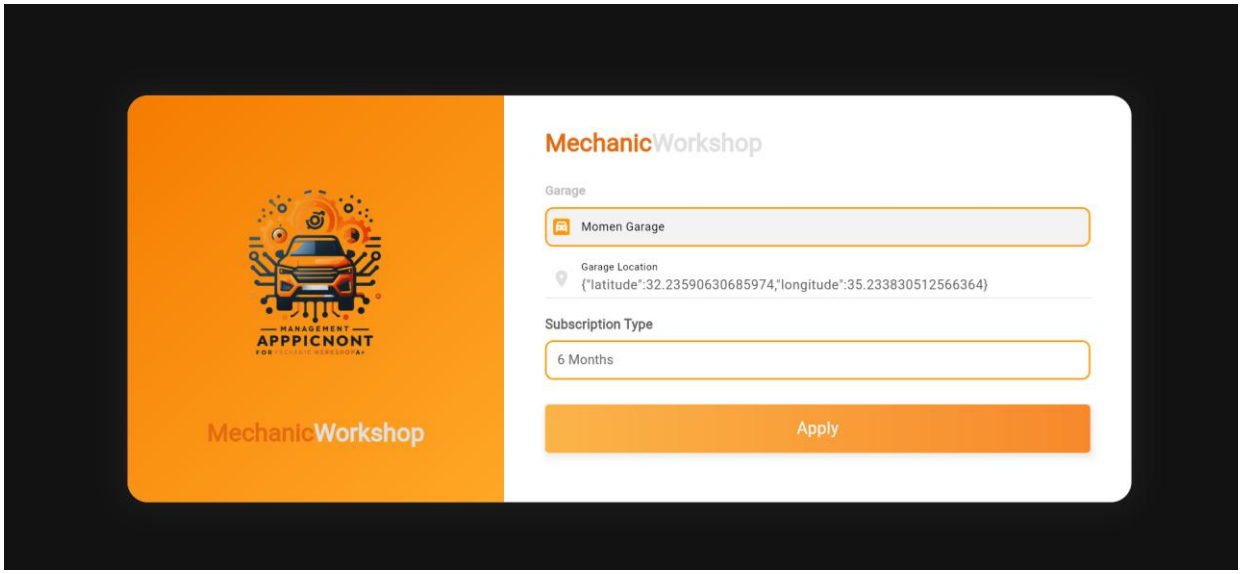
- Change Name


Account
Change Language
Mode
Home
Choose Language
العربية
اللغة العربية
English
English Language
Cancel
Save Changes

5.9.5.11 *Notification*

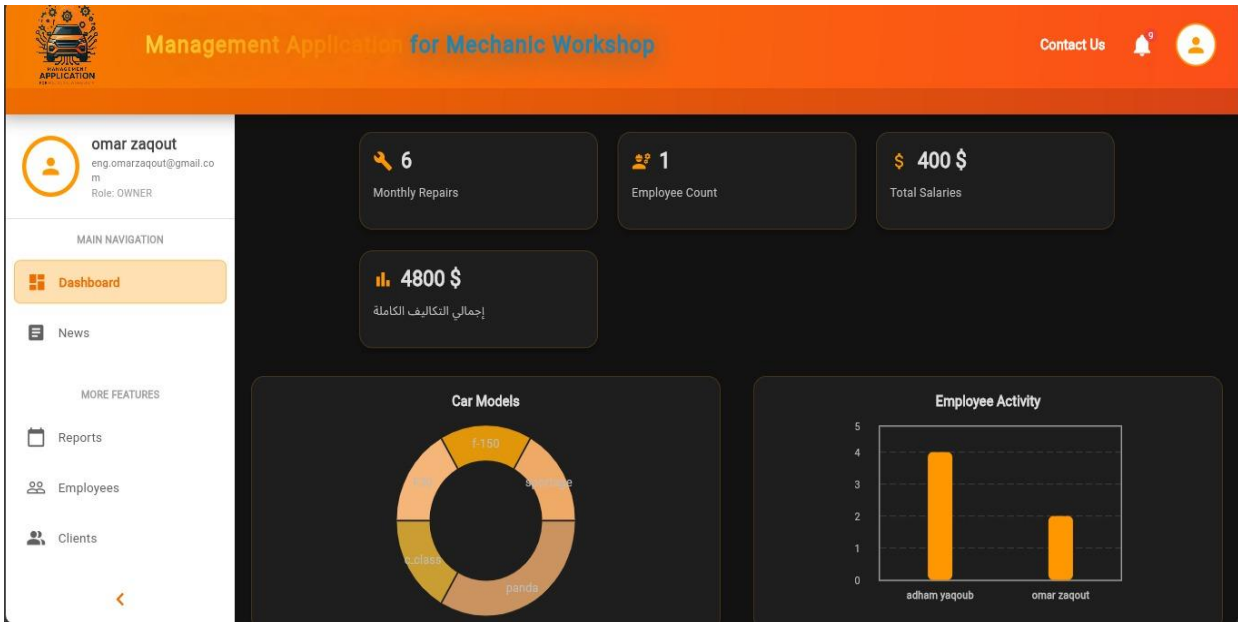


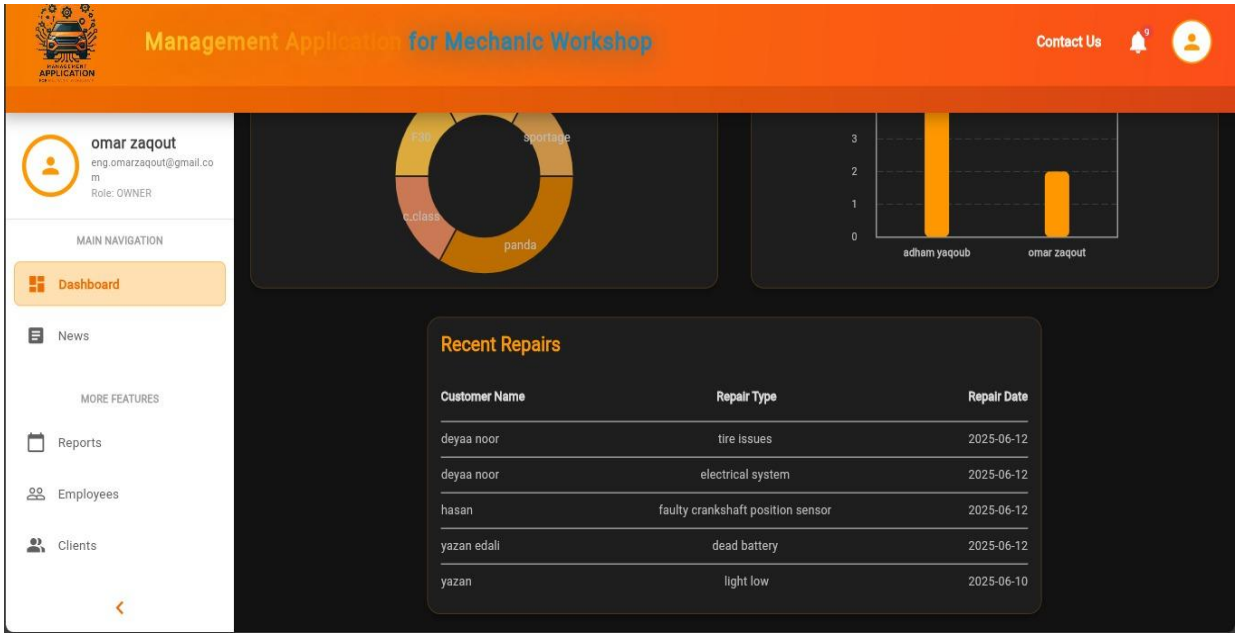
5.9.6 Apply Request Page



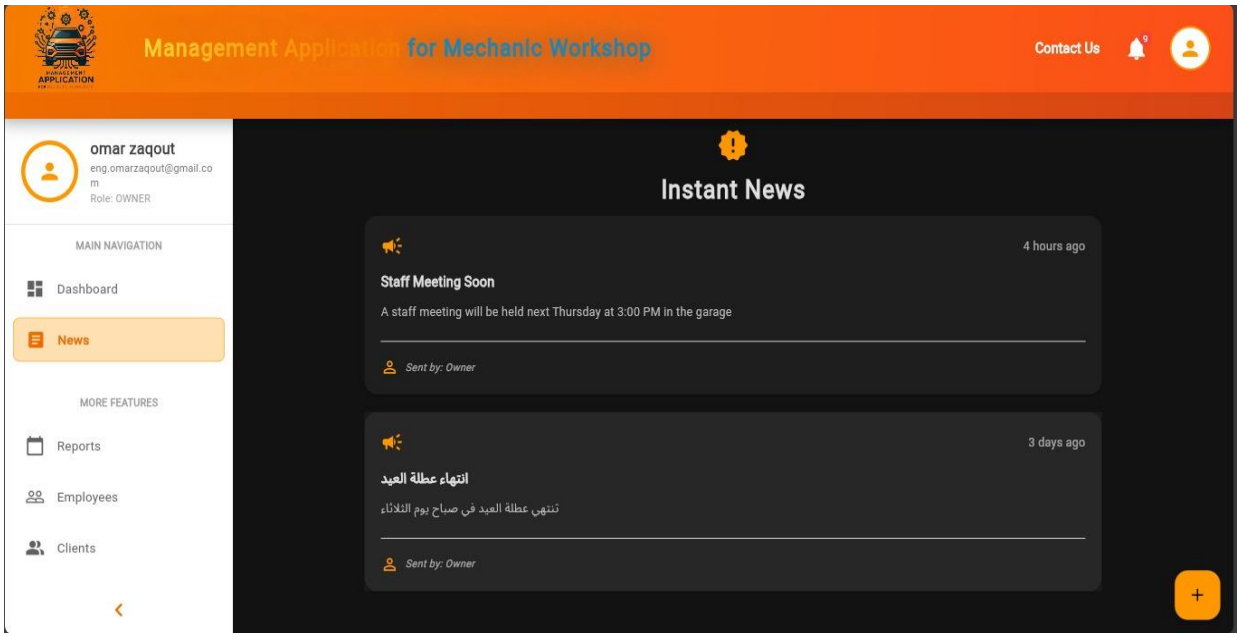
5.9.7 Owner Pages

5.9.7.1 *Dashboard screen*

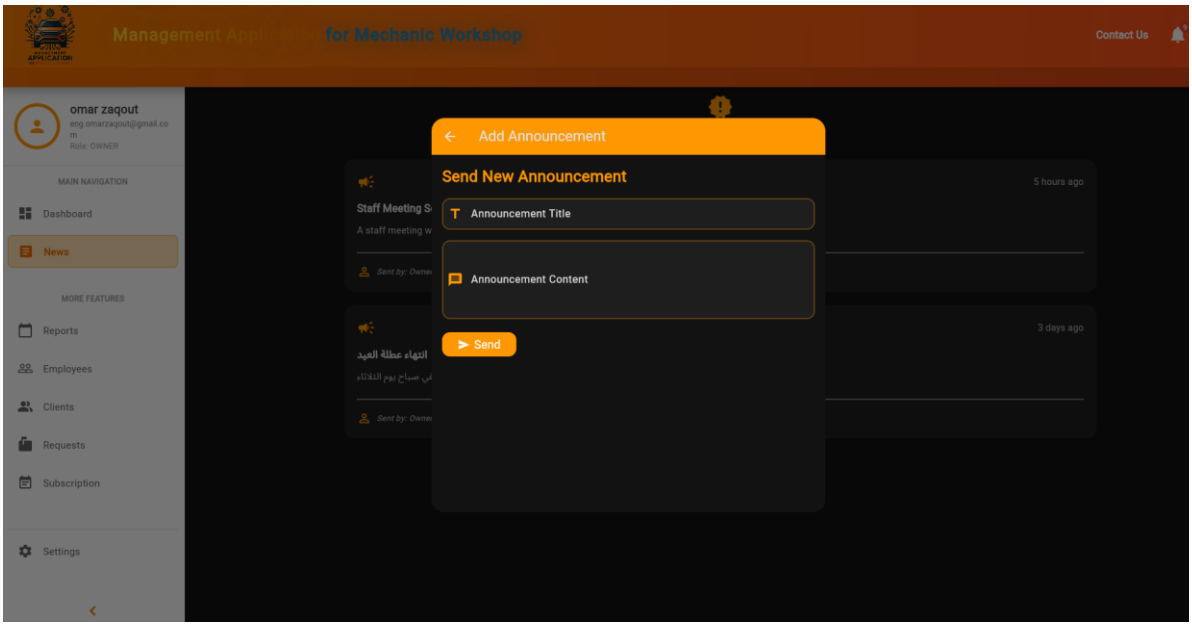




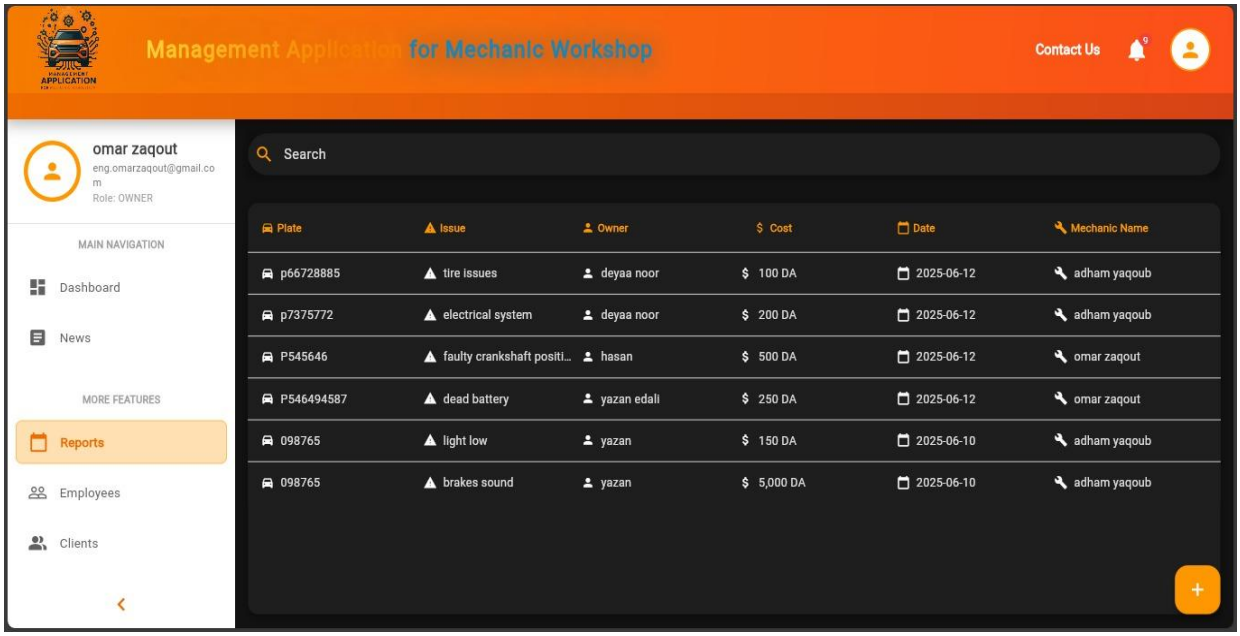
5.9.7.2 *News Screen*



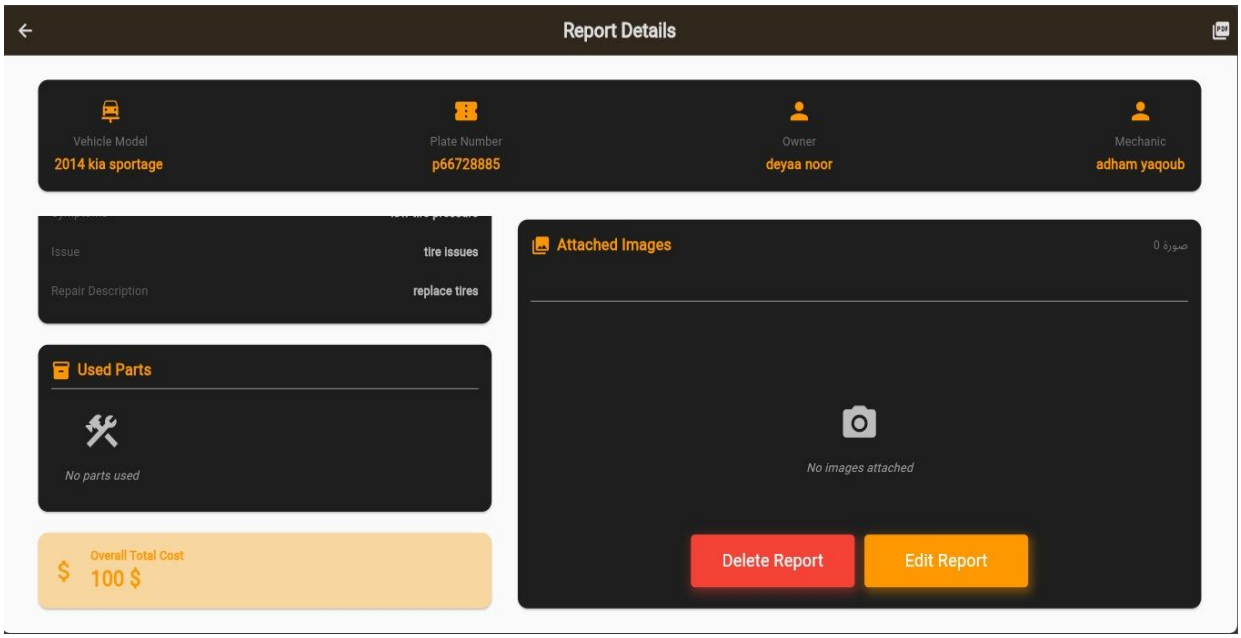
- Add News



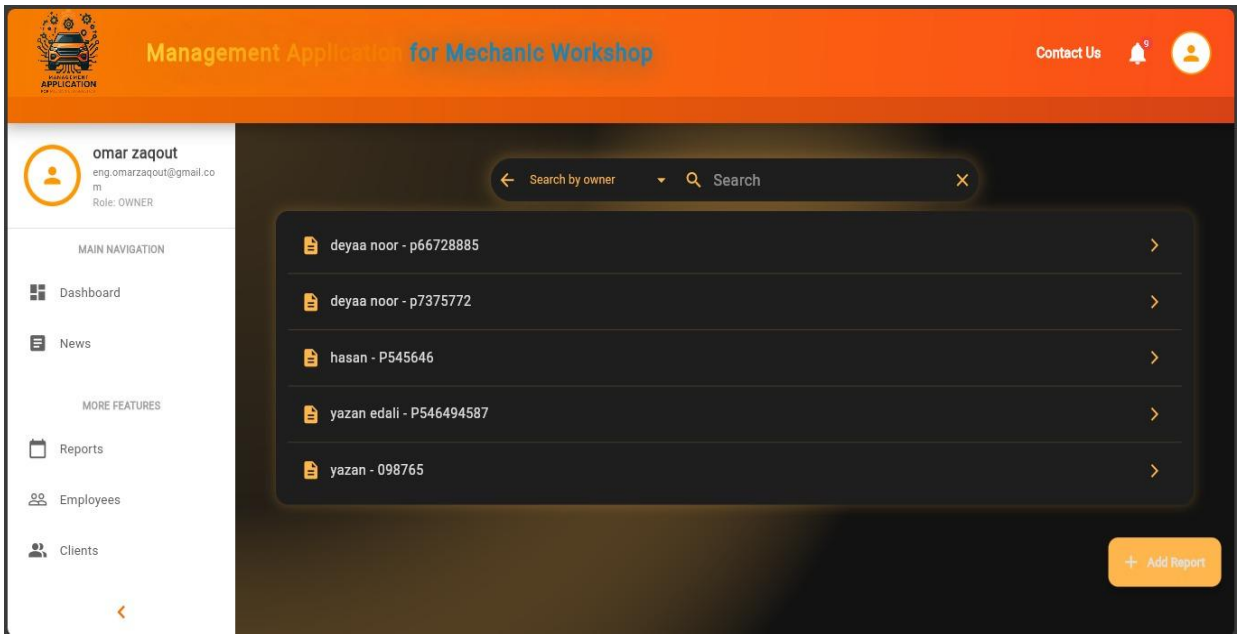
5.9.7.3 *Report Screen*



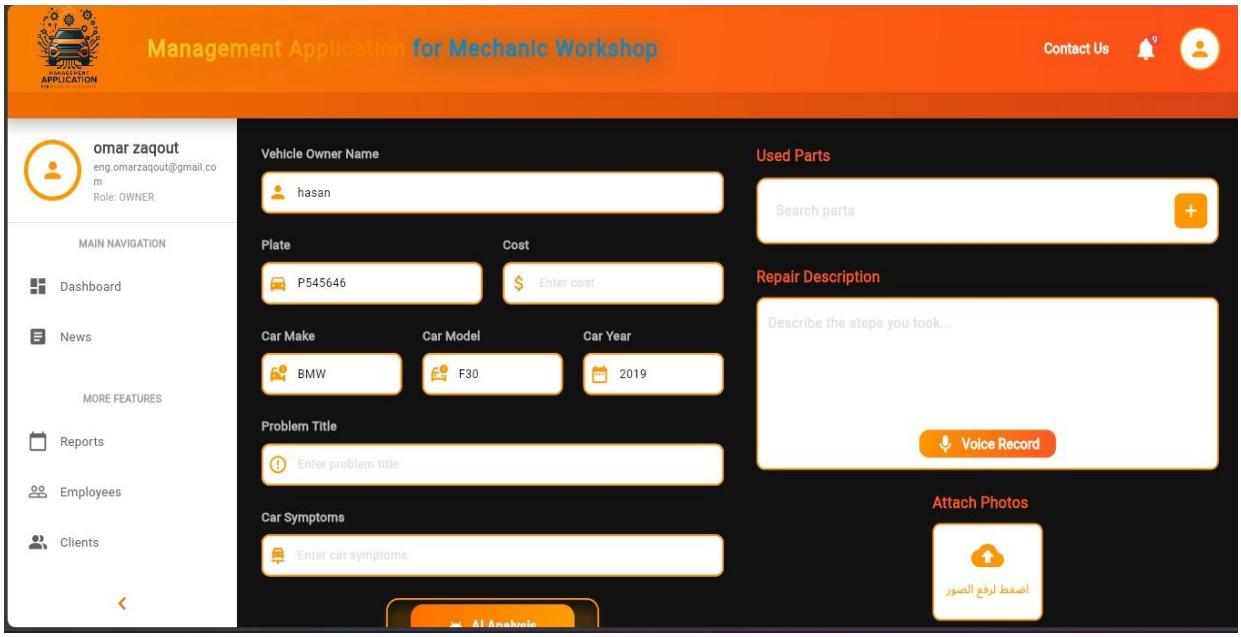
- Report Details



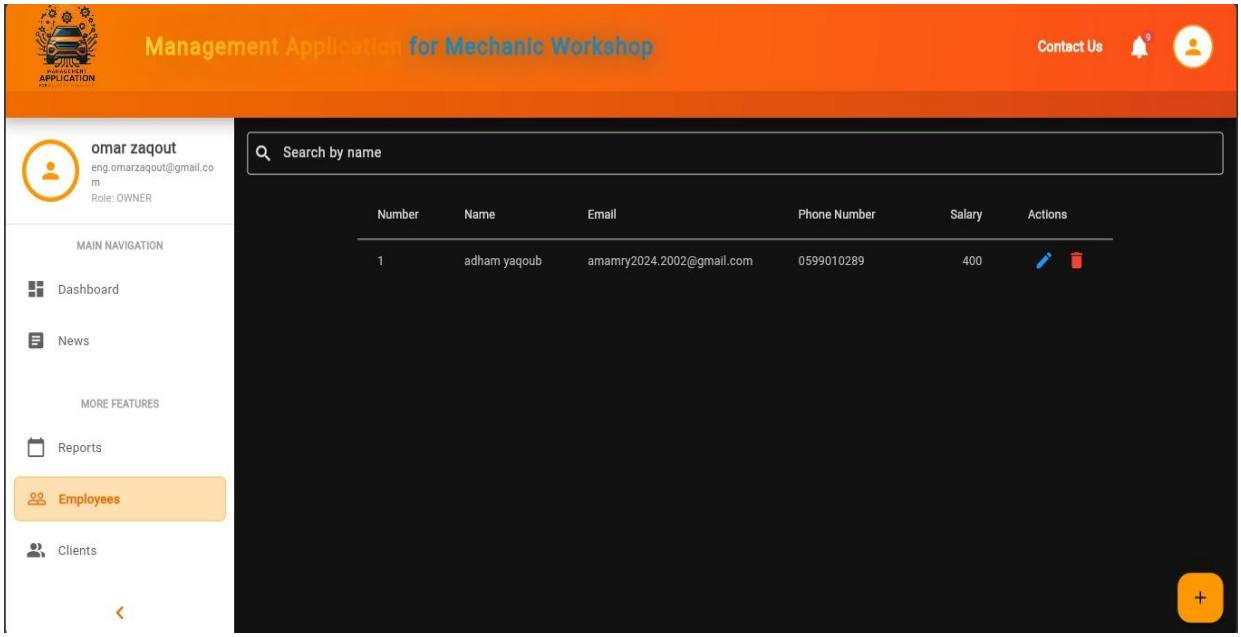
- Reporting records



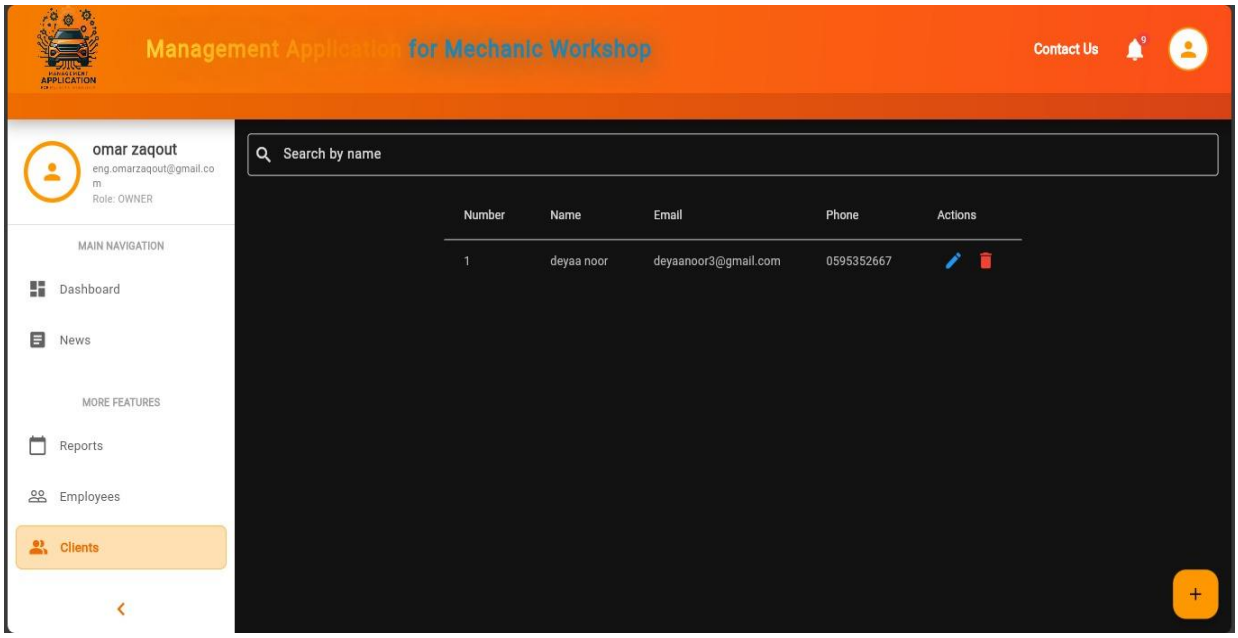
- Add a report for a previously registered user



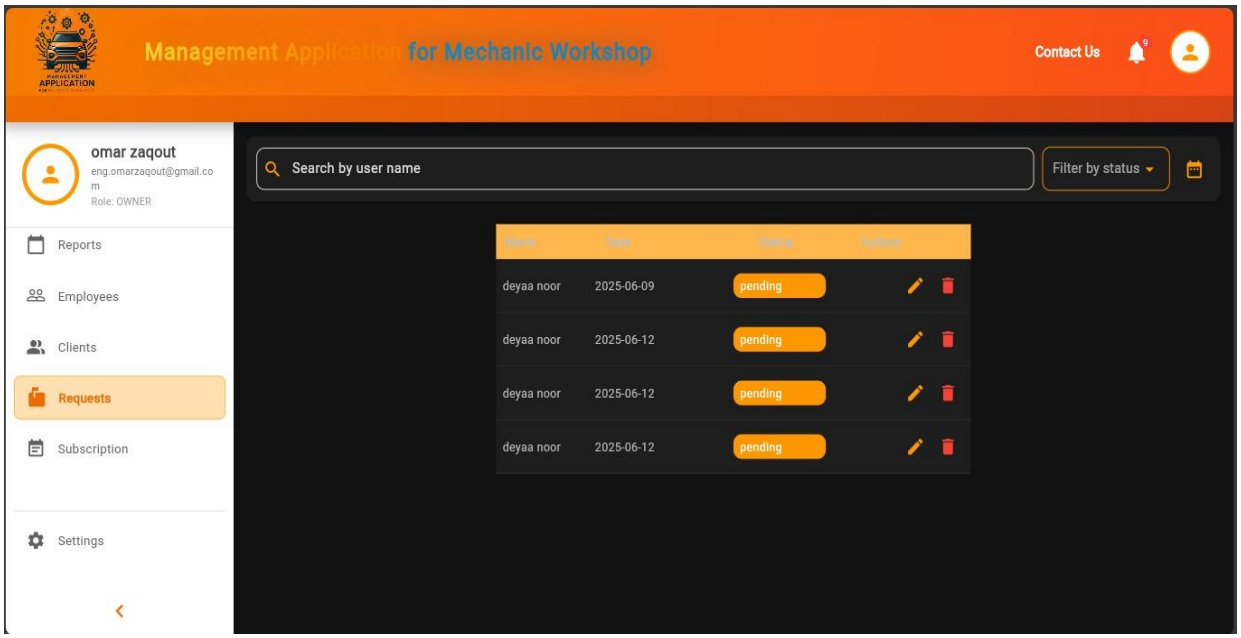
5.9.7.4 *Employee Screen*



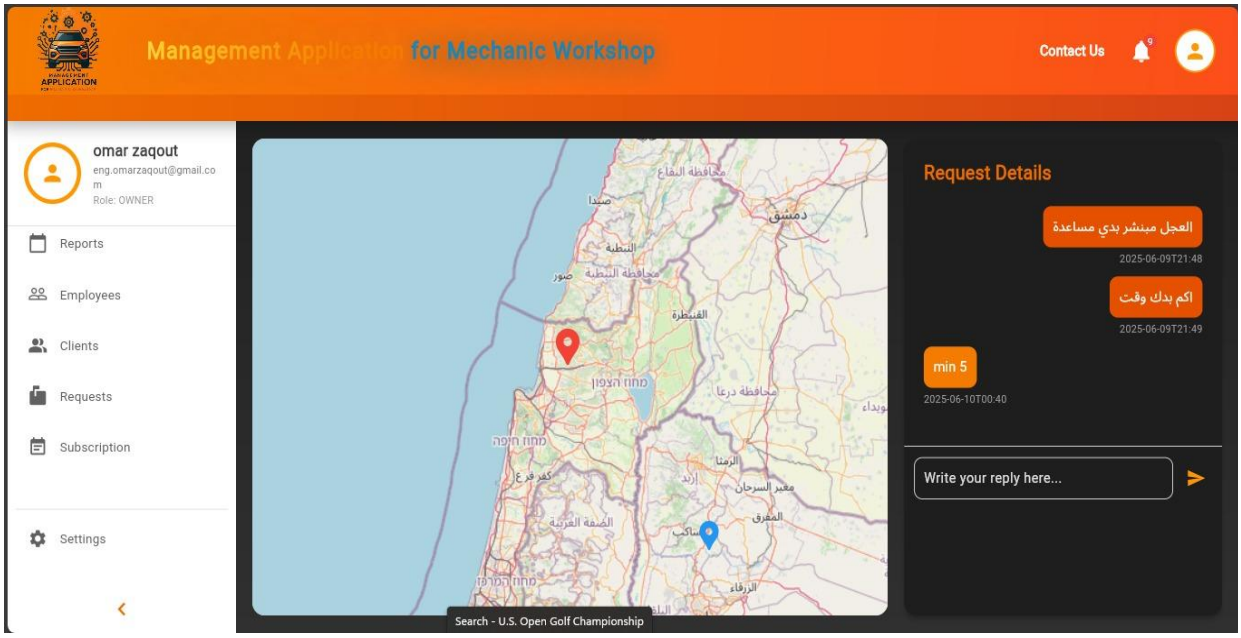
5.9.7.5 *Client Screen*



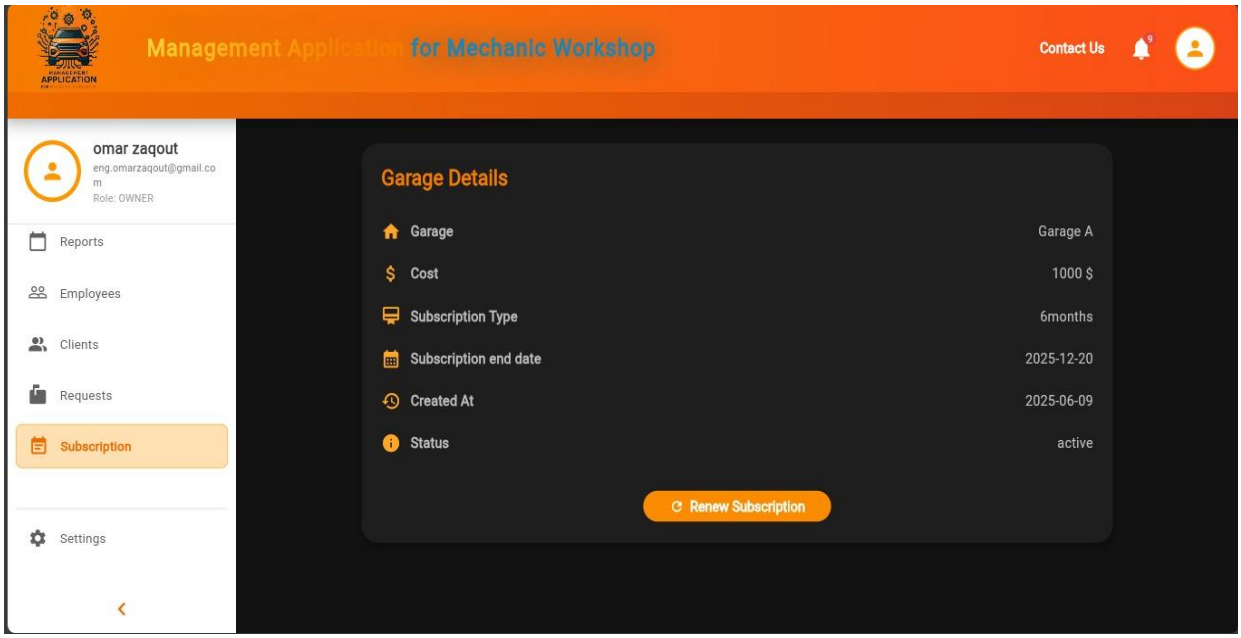
5.9.7.6 *Request Screen*



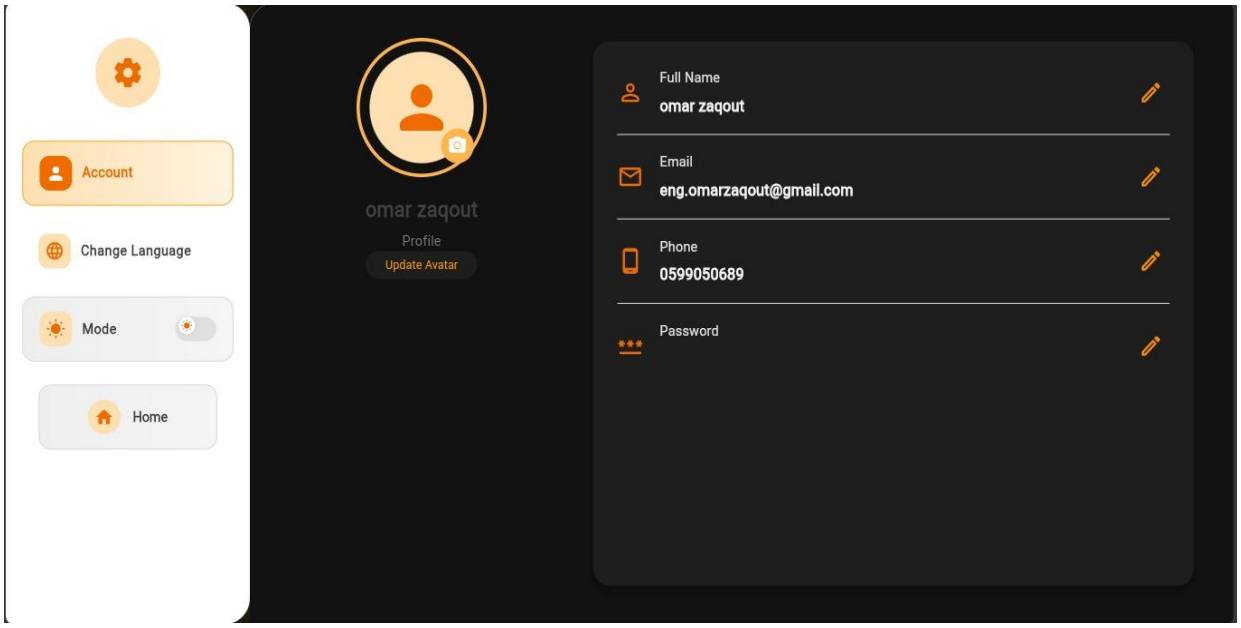
- Request Details



5.9.7.7 *Subscription Screen*

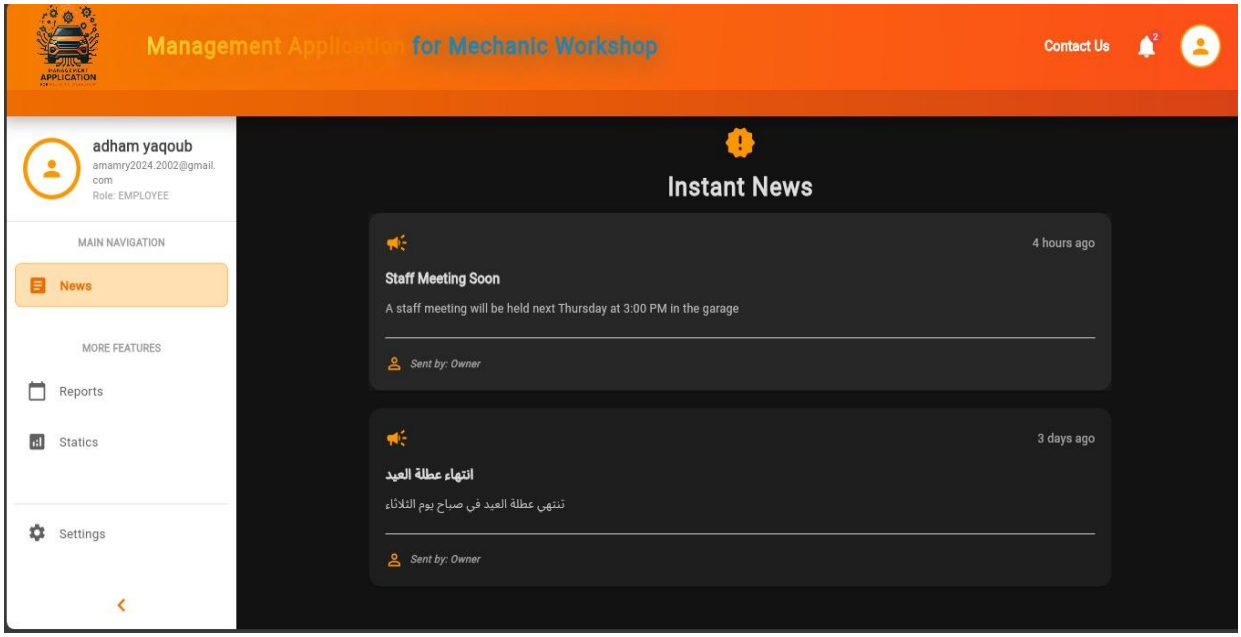


5.9.7.8 *Settings Screen*

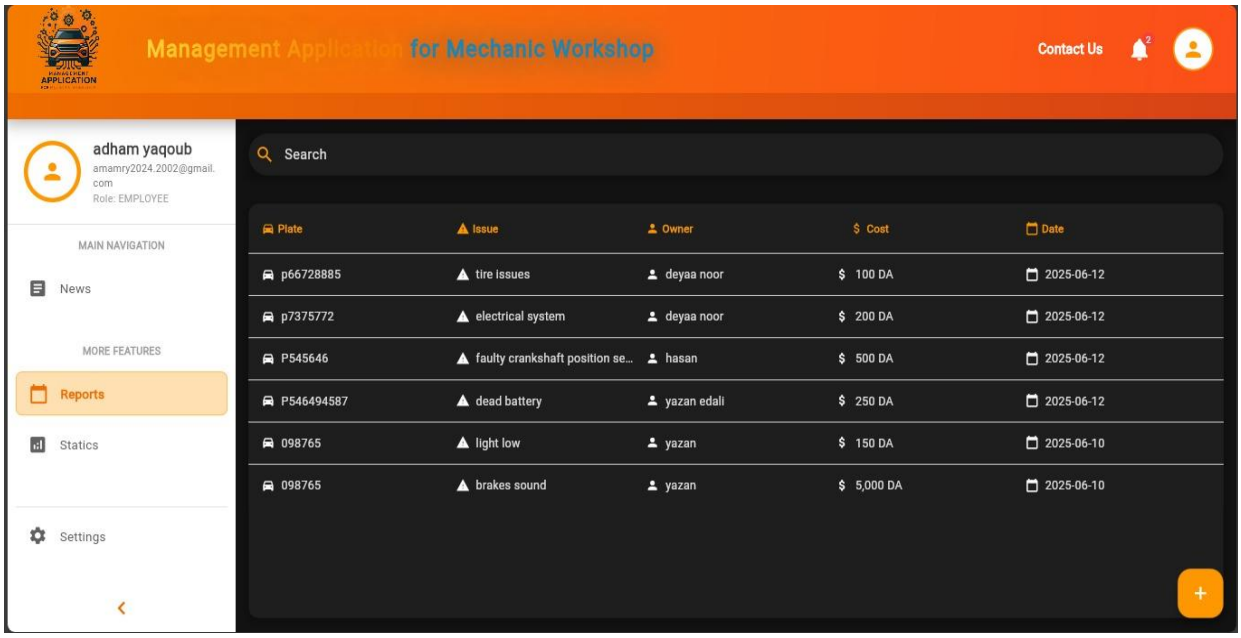


5.9.8 Employee Page

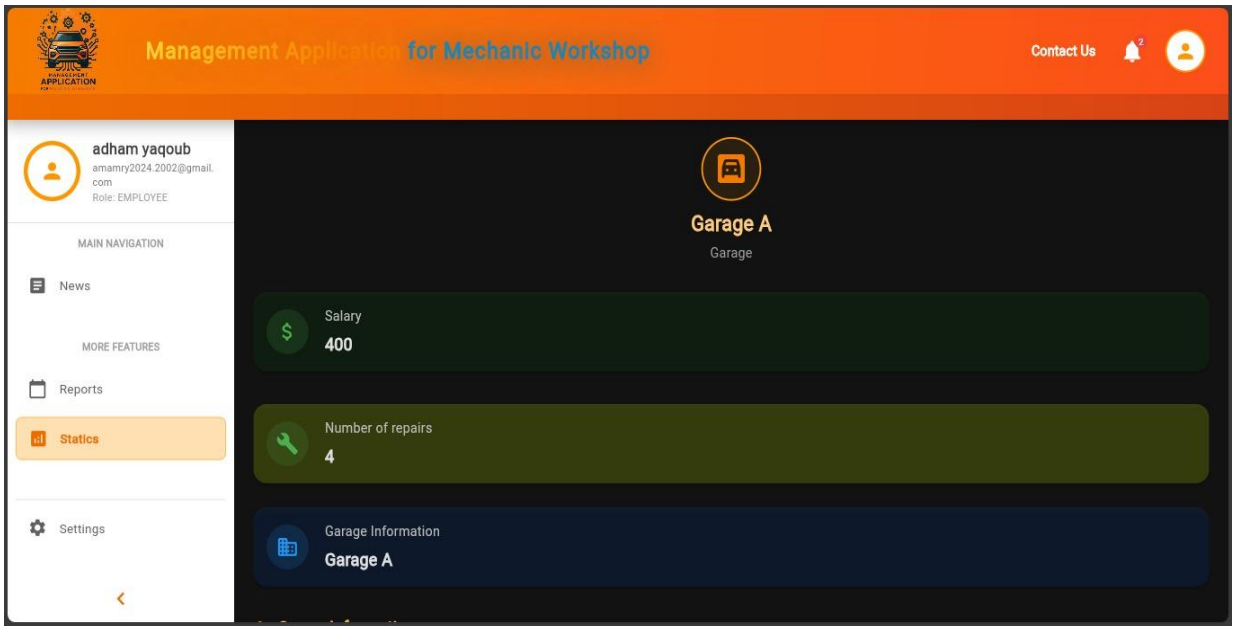
5.9.8.1 *News Screen*

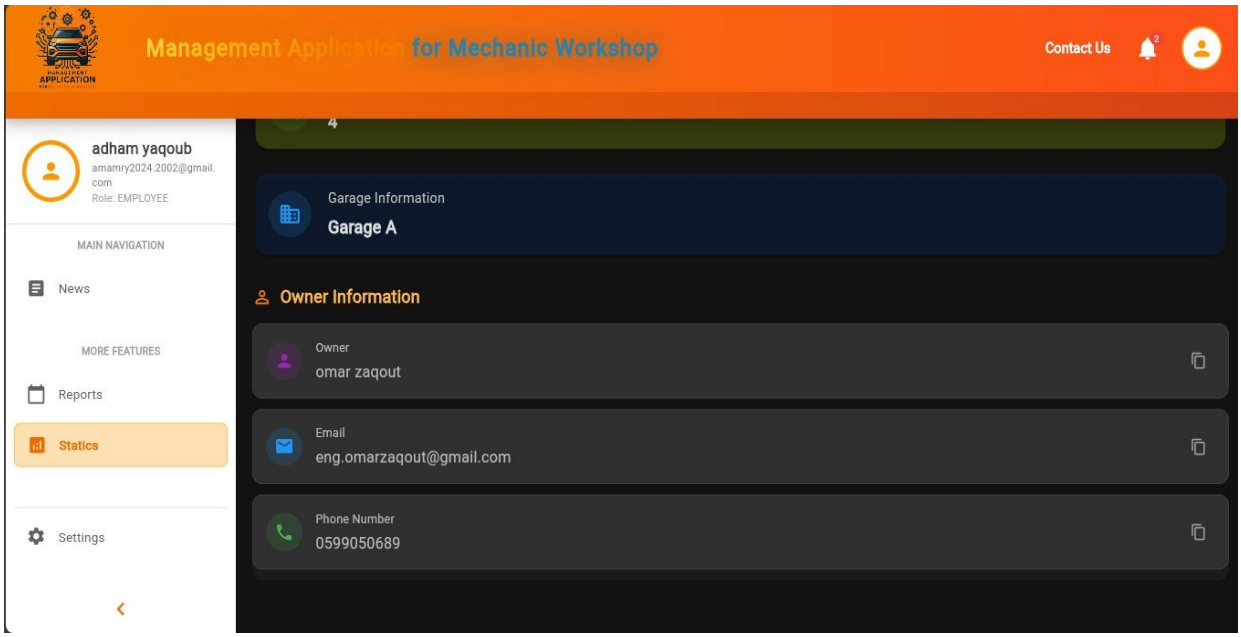


5.9.8.2 *Report screen*



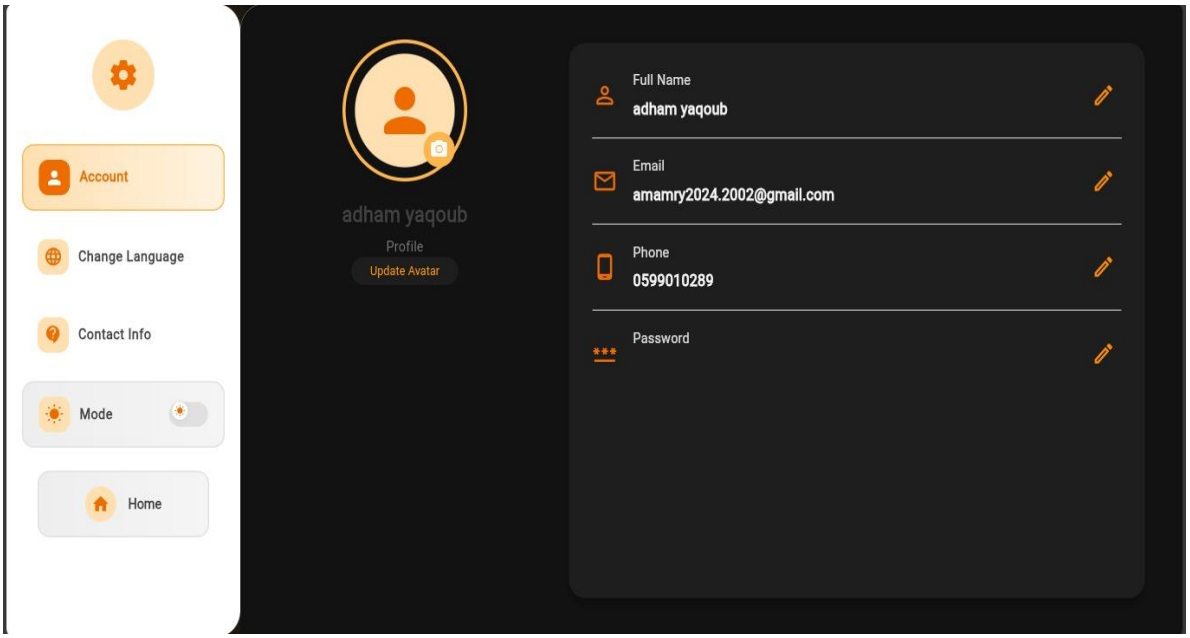
5.9.8.3 *Statics*



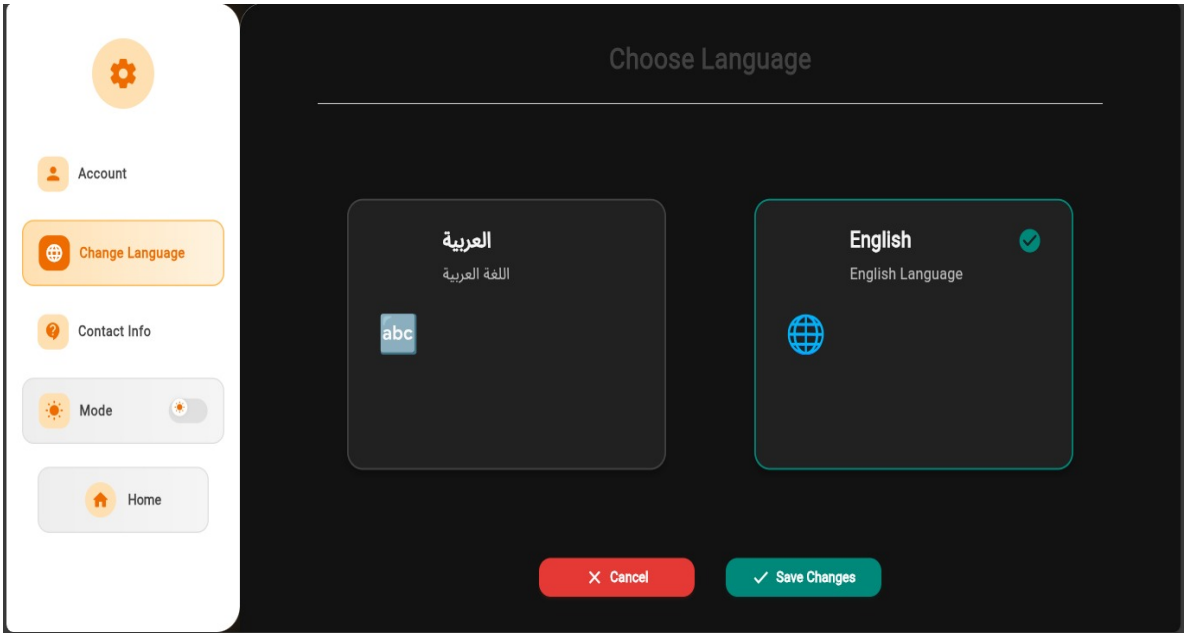


5.9.8.4 *Settings Screen*

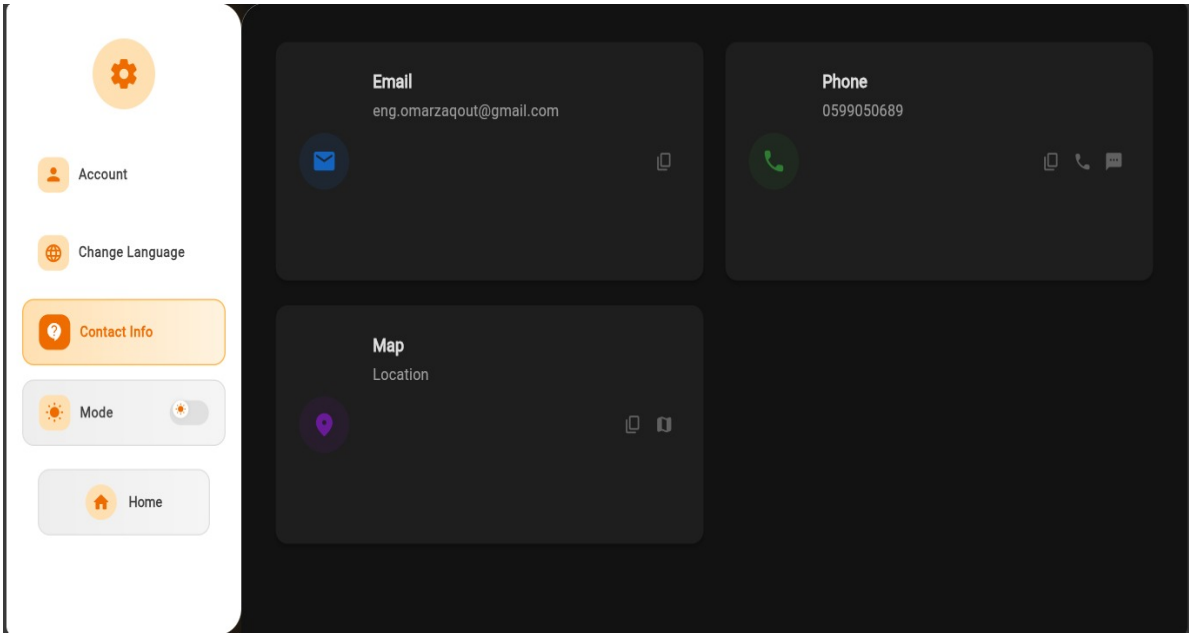
- Account



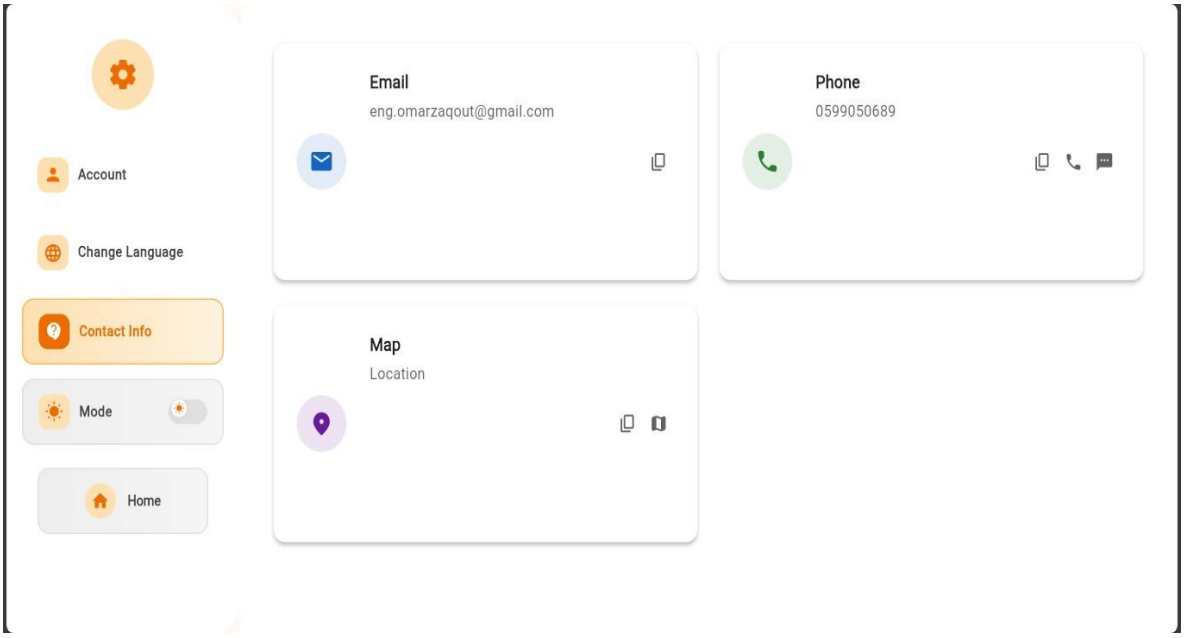
- Change Language



- Contact info

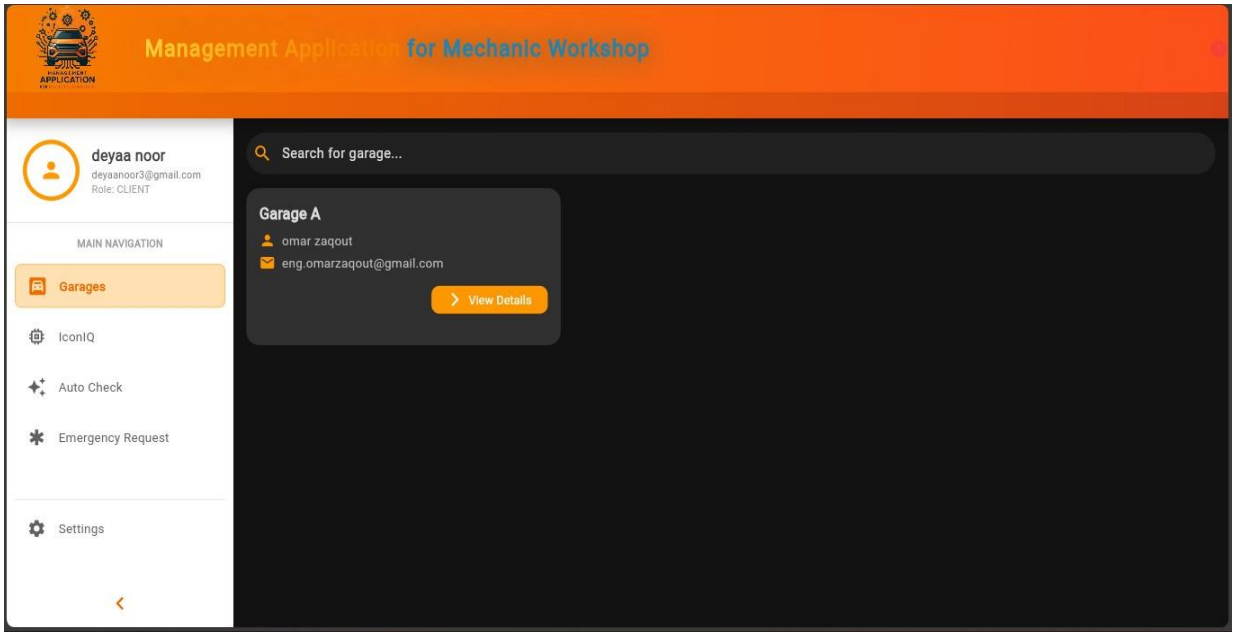


- Light Mood

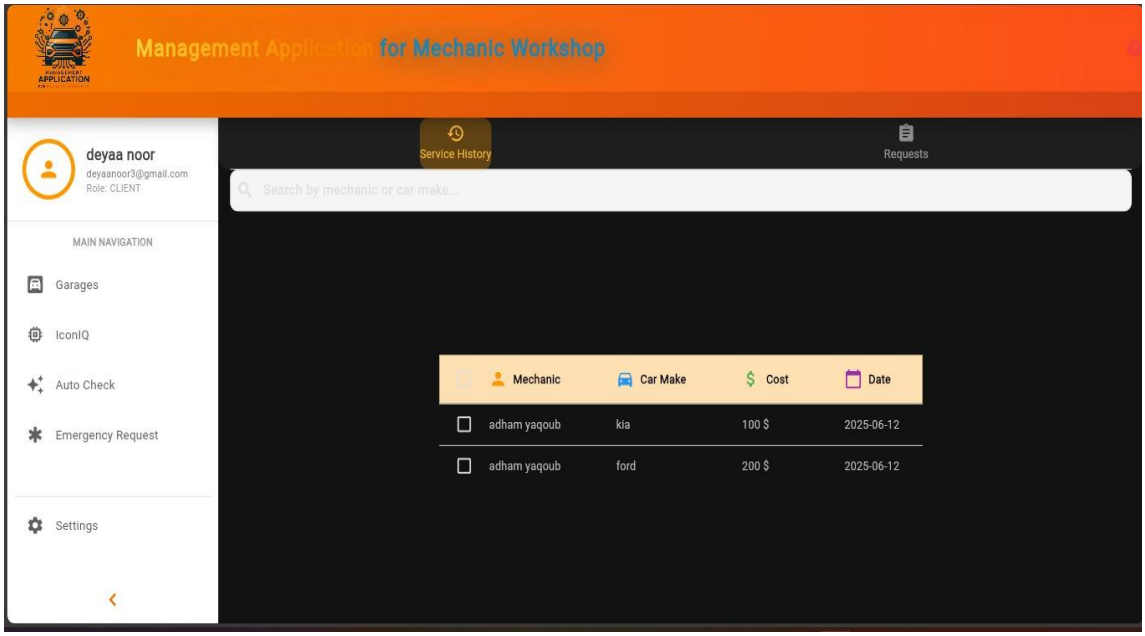


5.9.9 Client Page

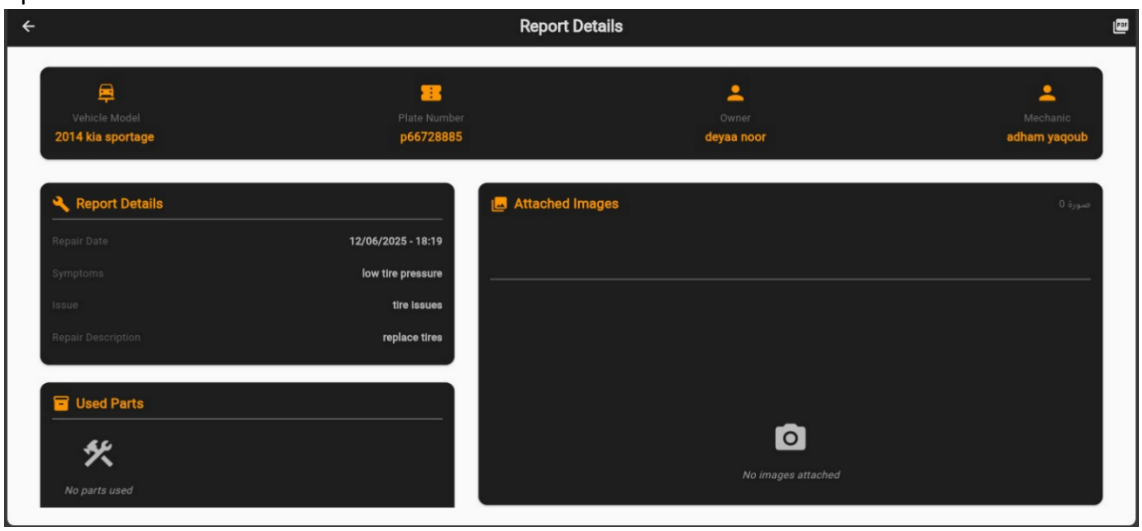
5.9.9.1 *Garages*

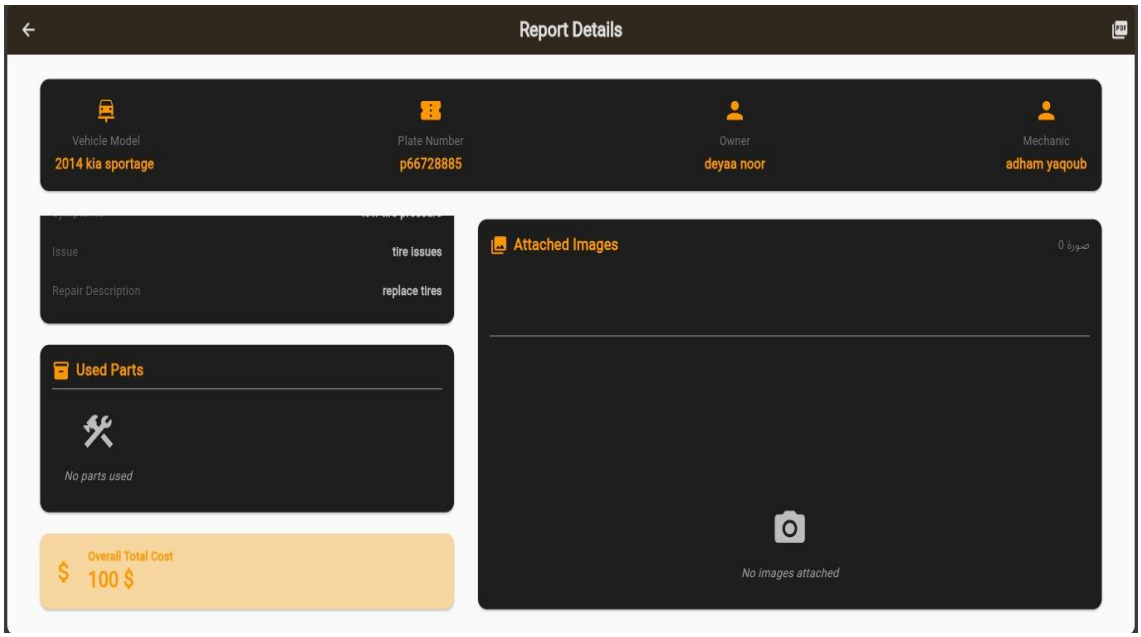


- Garage A

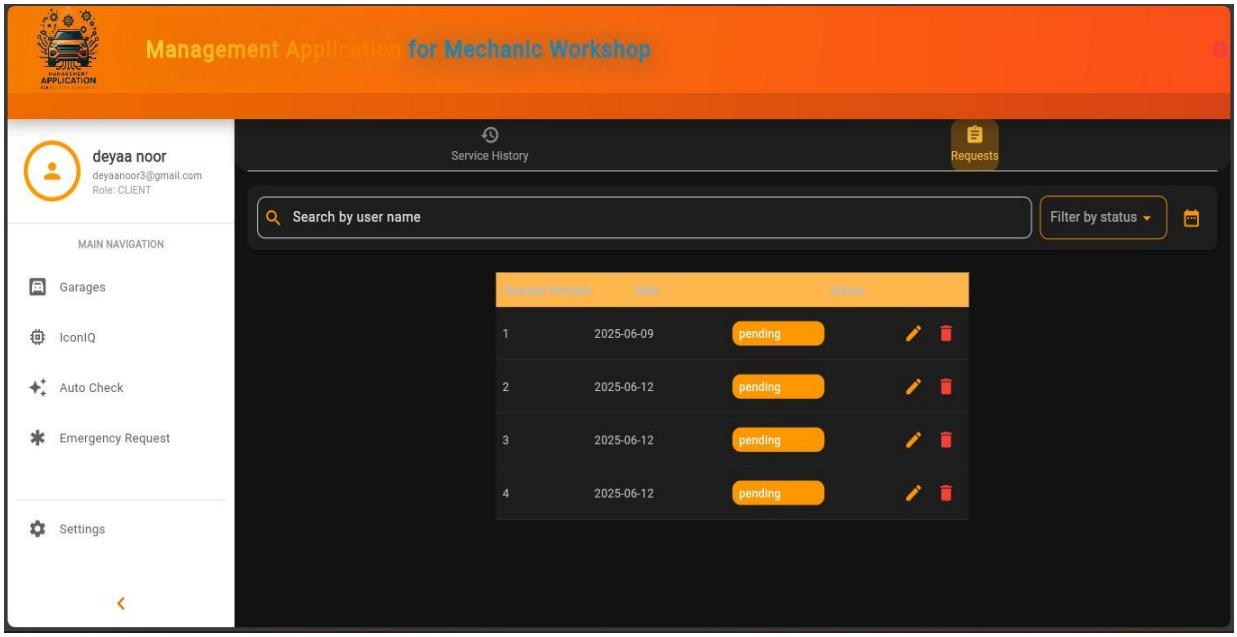


- Report Details

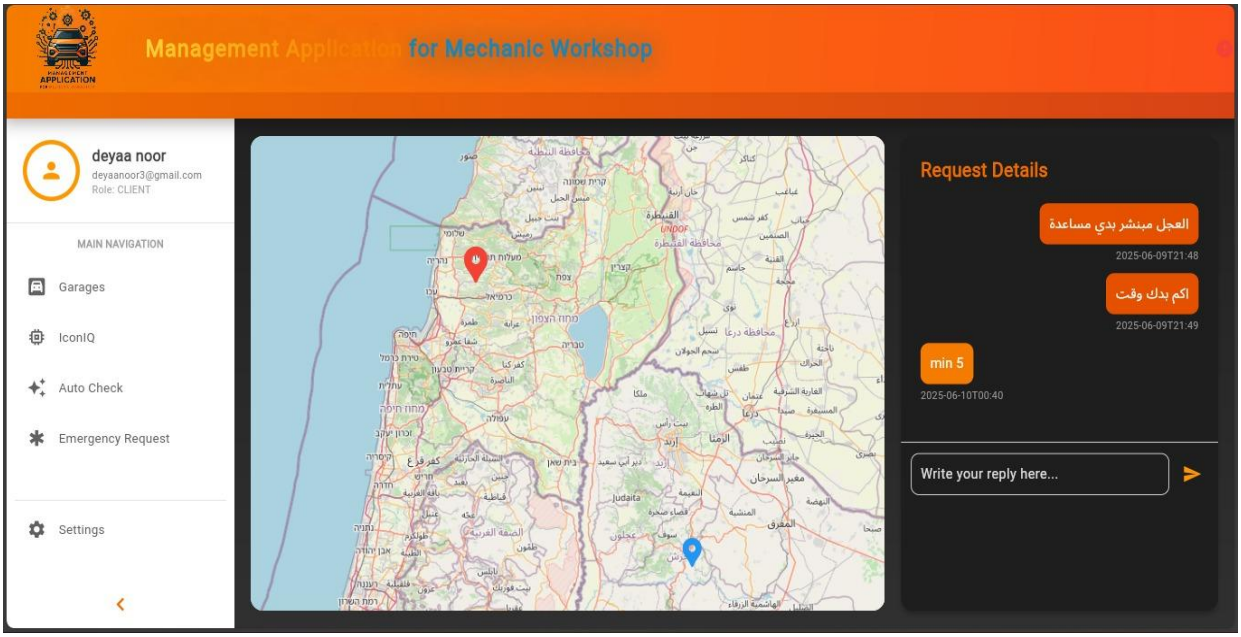




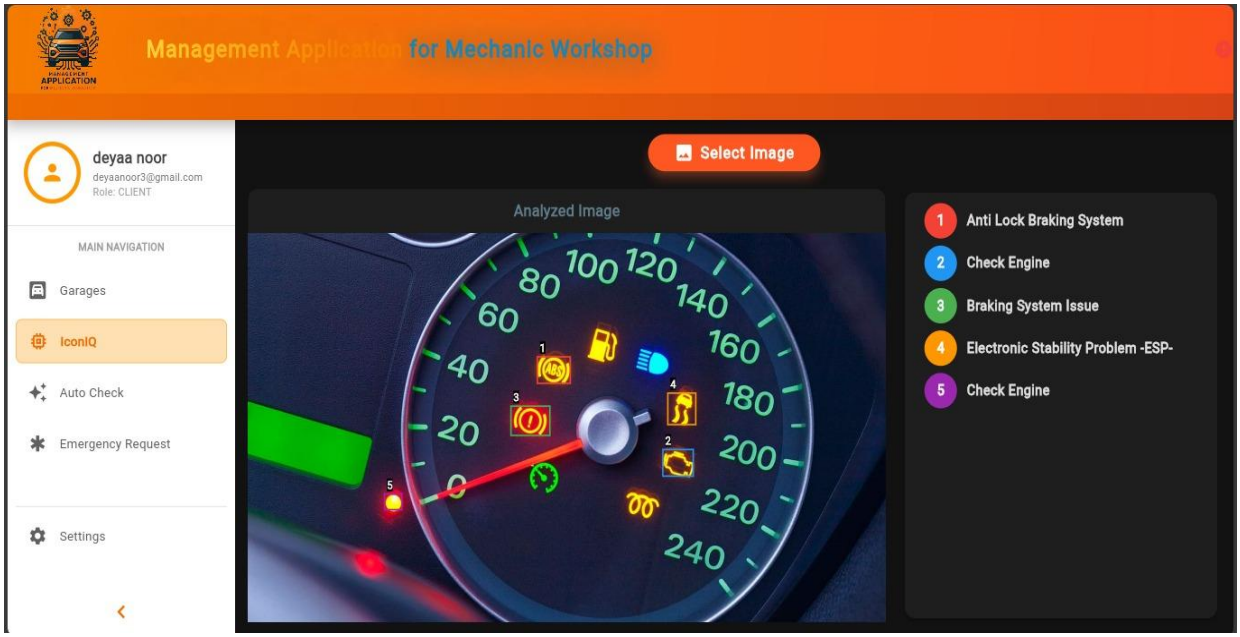
5.9.9.2 *Request Screen*



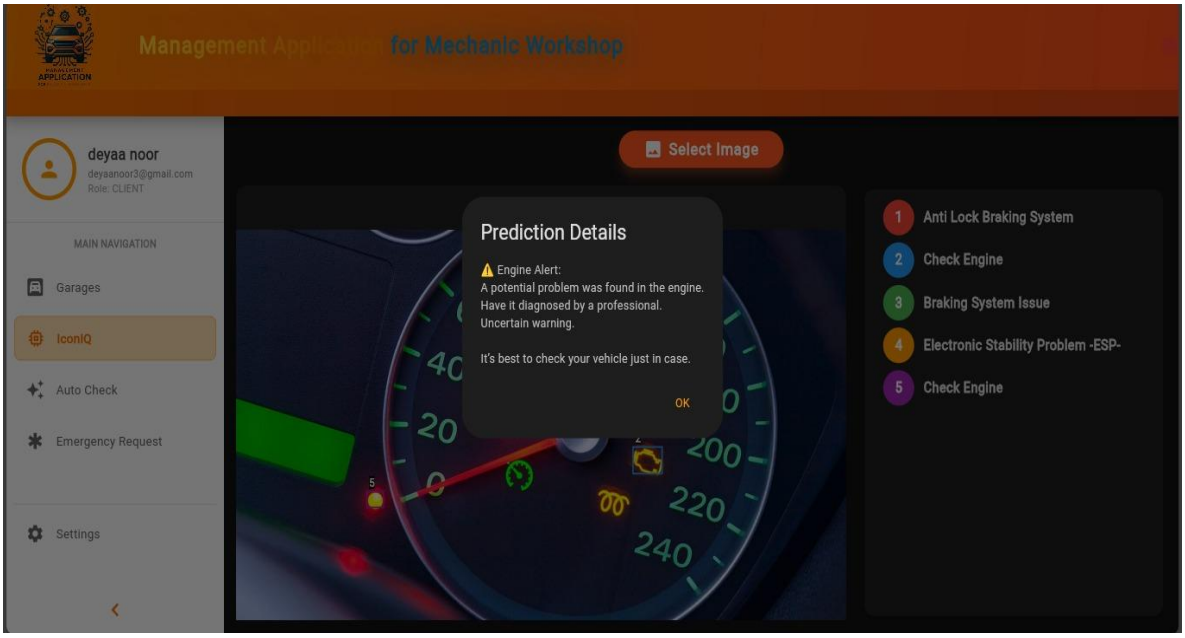
- Request Details


Management Application for Mechanic Workshop
deyaa noor
deyaanoor3@gmail.com
Role: CLIENT
MAIN NAVIGATION
Garages
IconIQ
Auto Check
Emergency Request
Settings
Request Details
العجل مبنشر بدي مساعدة
2025-06-09T21:48
اكم بدك وقت
2025-06-09T21:49
min 5
2025-06-10T00:40
Write your reply here...

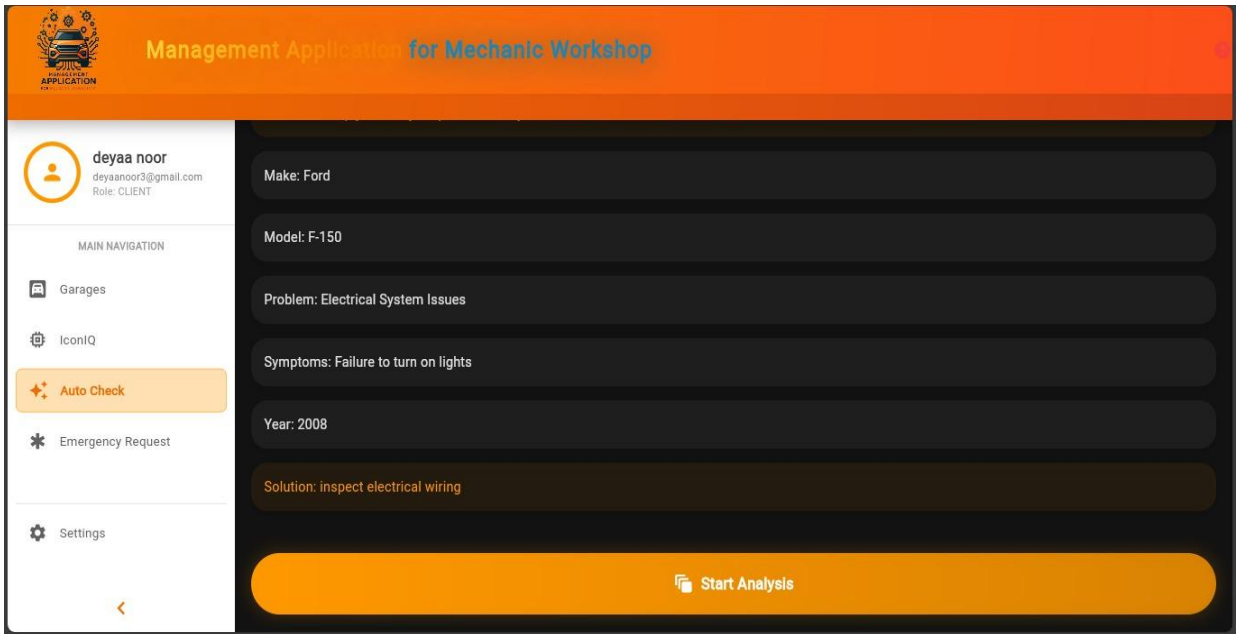
5.9.9.3 *Icon IQ*

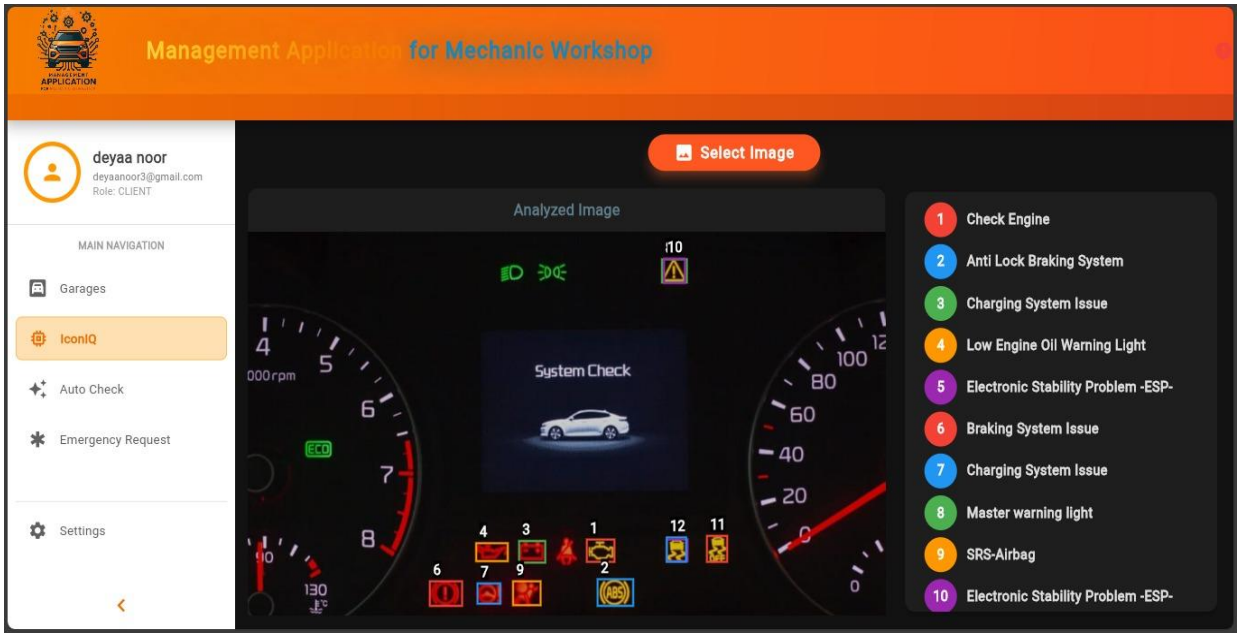

Management Application for Mechanic Workshop
deyaa noor
deyaanoor3@gmail.com
Role: CLIENT
MAIN NAVIGATION
Garages
IconIQ
Auto Check
Emergency Request
Settings
Select Image
Analyzed Image
1 Anti Lock Braking System
2 Check Engine
3 Braking System Issue
4 Electronic Stability Problem -ESP-
5 Check Engine

- Prediction details

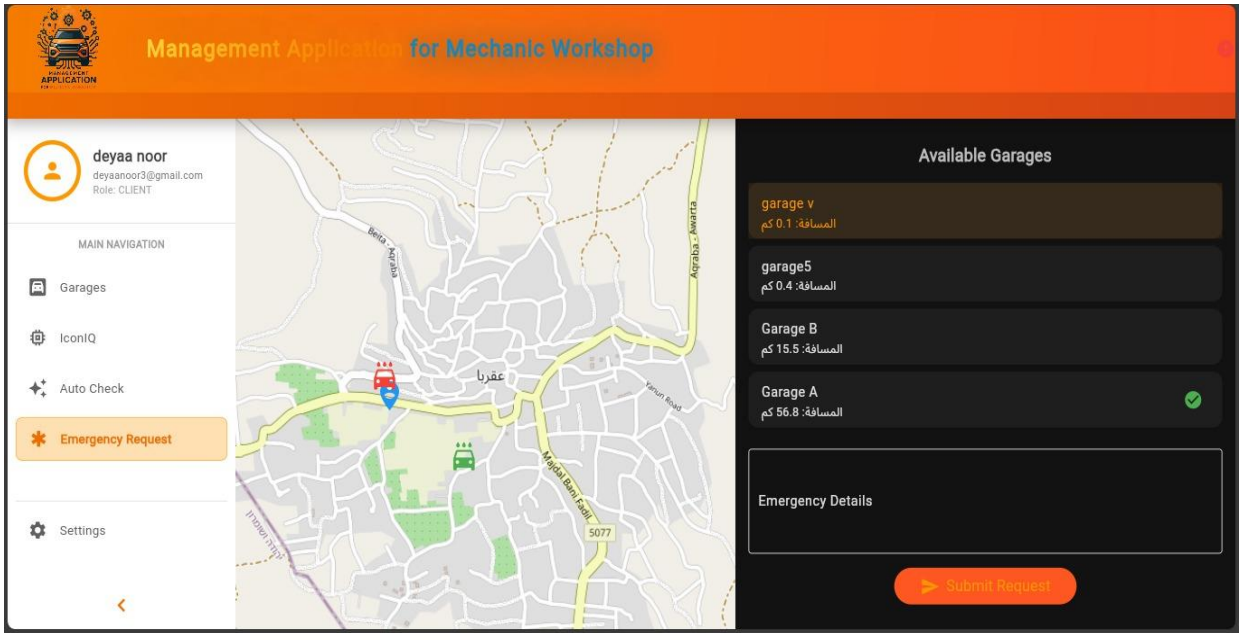


5.9.9.4 *Auto Check*

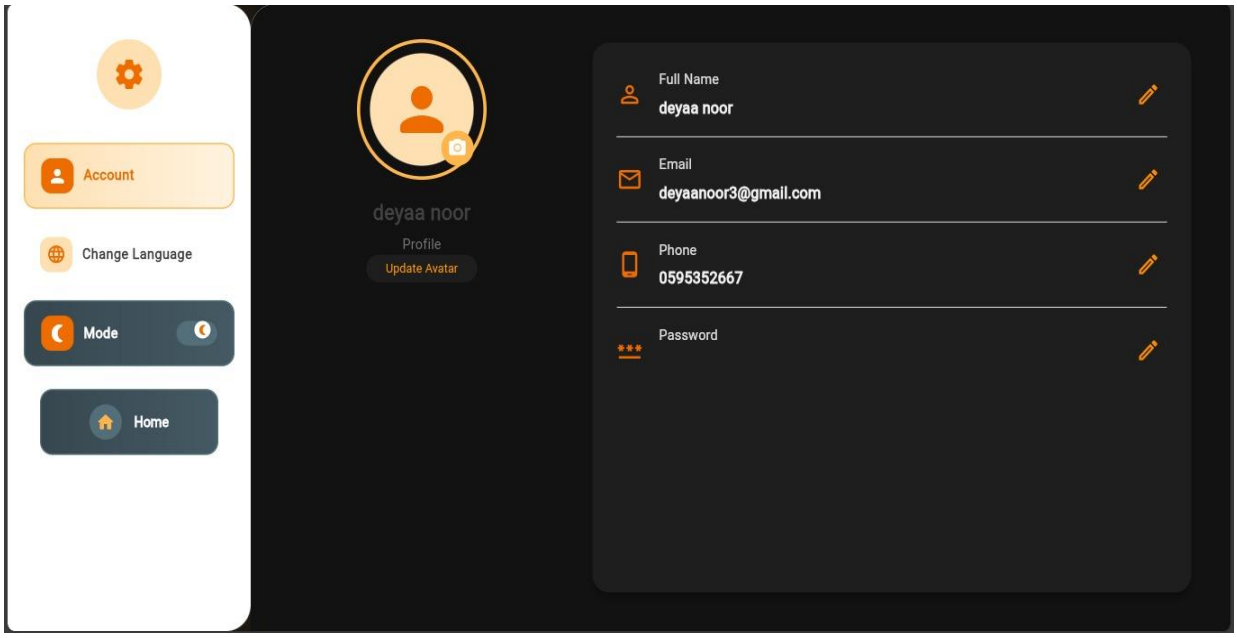




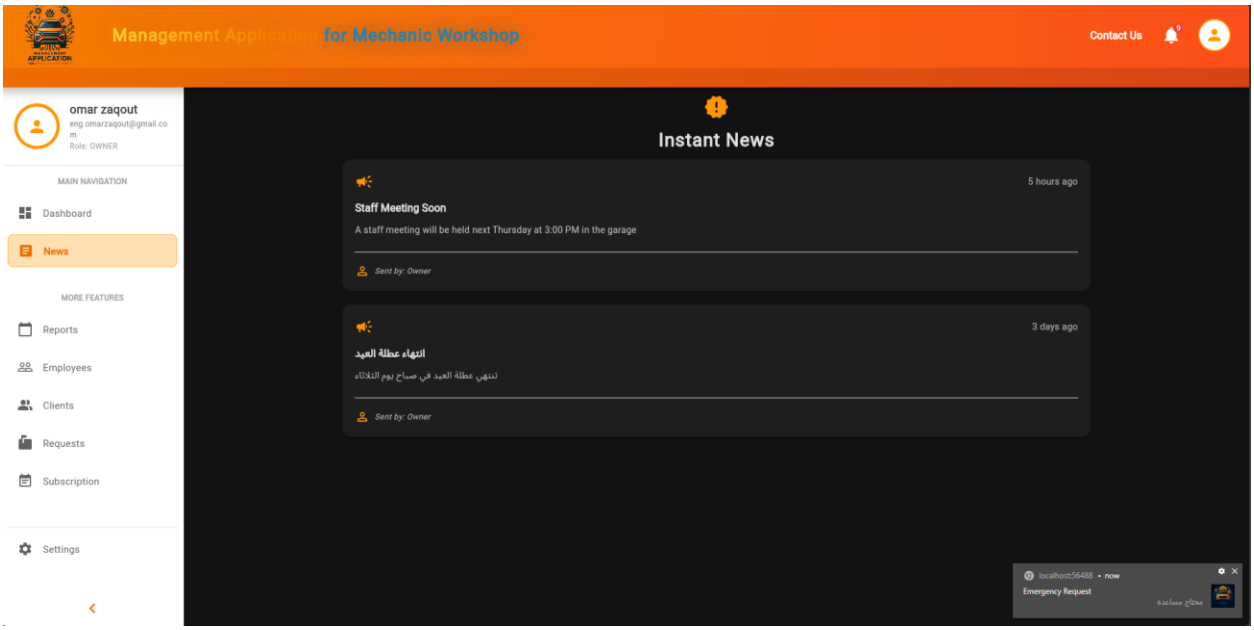
5.9.9.5 *Emergency Request*



5.9.9.6 *Settings*



notification



6 Conclusion

In this project, we developed a smart system designed to improve the management and efficiency of car repair workshops. The system aimed to streamline operations, enhance communication between different users (administrators, technicians, and customers), and integrate smart features that bring real value to the workshop experience.

Throughout the project, we successfully implemented the core functionalities, including user management, service tracking, and real-time communication. Our solution was built using modern technologies to ensure scalability, performance, and user-friendliness.

Despite facing challenges related to data integration and real-time system response, we were able to overcome them with teamwork and continuous learning.

In conclusion, the project achieved its main goals and proved to be a valuable step toward smarter workshop management. Future work could include the integration of machine learning for predictive maintenance, improved reporting features, and a more advanced mobile experience for users.

7 References

[1] Q. H. Mahmoud, Cross-Platform Mobile Application Development with Flutter, Berlin: Springer, 2020.

[2] K. Chodorow, MongoDB: The Definitive Guide, Sebastopol: O'Reilly Media, 2013.

[3] M. H. T. H. N. R. M. Cantelon, Node.js in Action, Shelter Island: Manning Publications, 2014.

[4] "Render Documentation," Render, [Online]. Available: https://render.com/docs.

[5] M. A. e. al, "TensorFlow: A system for large-scale machine learning," in *OSDI*, 2016.

[6] "Roboflow Documentation," Roboflow, [Online]. Available: https://roboflow.com/.

[7] D. L. D. Wang, "A Deep Learning Framework for Vehicle Fault Diagnosis Using Sensor Data," *Sensors,* vol. 20, no. 20, p. 5686, 2020.

[8] A. B. A. Kumar, "AI-Driven Vehicle Diagnostic Systems: A Review," *Journal of Intelligent Transportation Systems,* vol. 26, no. 4, pp. 341-353, 2021.

[9] Firebase, "Firebase Cloud Messaging (FCM) Documentation," Google, [Online]. Available: https://firebase.google.com/docs/cloud-messaging. [Accessed 2025].