Project Report

**on**

### ON-ROAD VEHICLE BREAKDOWN ASSISTANCE

### by

### 

**ON-ROAD VEHICLE BREAKDOWN ASSISTANCE**

**ABSTRACT**

This vehicle management system is a fully customized desktop application where company staff can view each customer’s orders and give a solution to those vehicle problems. The locating system allows you to search for the mechanics from different locations. Admin handles and can access user details as well as the mechanic details. Admin has the access to allow/block and view the mechanics. This online mechanic locator reduces your work and easily finds the mechanics from various areas. Reduce your time and cost.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT ACKNOWLEDGEMENT LIST OF TABLES**  **LIST OF FIGURES** | **i iii vii**  **vii** |
| **CHAPTERS** | **TITLE** | **PAGE NO.** |
| **1** | **INTRODUCTION**  1.1 PROJECT INTRODUCTION | 6 |
| **2** | **WORKING ENVIRONMENT**   * 1. HARDWARE REQUIREMENT   2. SOFTWARE REQUIREMENT   3. SYSTEM SOFTWARE | 7  7  8 |
| **3** | **SYSTEM ANALYSIS**   * 1. FEASIBILITY STUDY   2. EXISTING SYSTEM   3. DRAWBACKS OF EXISTING SYSTEM   4. PROPOSED SYSTEM   5. BENEFITS OF PROPOSED SYSTEM   6. SCOPE OF THE PROJECT | 16  18  18  19  19  20 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **4** | **SYSTEM DESIGN**   * + DATA FLOW DIAGRAM   + USE CASE DIAGRAM   + CLASS DIAGRAM   + ARCHITECTURE DIAGRAM   + SEQUENCE DIAGRAM   + DATABASE DESIGN | 21  23  24  25  26  27 |
| **5** | **PROJECT DESCRIPTION**   * 1. OBJECTIVE   2. MODULE DESCRIPTION   3. IMPLEMENTATION | 30  31  33 |
| **6** | **SYSTEM TESTING**   * + TESTING DEFINITION   + TESTING OBJECTIVE   + TYPES OF TESTING | 34    34  35 |
| **7** | **CONCLUSION**   * + SUMMARY   + FUTURE ENHANCEMENTS | 36  36 |
| **8** | **APPENDIX**   * + SCREENSHOTS   + CODING | 37  44 |
| **9** | **BIBLIOGRAPHY AND REFERENCES** | 50 |

**LIST OF FIGURES**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Fig. No** | **Figures Name** | **Page No.** |
| 1 | 4.1 | DATA FLOW DIAGRAM | 22 |
| 2 | 4.2 | USE CASE DIAGRAM | 23 |
| 3 | 4.3 | CLASS DIAGRAM | 24 |
| 4 | 4.4 | SEQUENCE DIAGRAM | 25 |
| 5 | 4.5 | ARCHITECTURE DIAGRAM | 26 |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **Tab. No.** | **Table Name** | **Page No.** |
| 1 | 4.6 | ADMIN TABLE | 27 |
| 2 | 4.7 | MECHANIC TABLE | 28 |
| 3 | 4.8 | USER TABLE | 29 |

**LIST OF TABLES**

**CHAPTER-1**

**INTRODUCTION**

**PROJECT INTRODUCTION**

The Road Assistance application was developed with the aim of providing emergency roadside assistance services round the clock to ensure a pleasurable and uninterrupted journey anywhere. The application is designed to enhance the user experience and ensure that users get immediate and hassle-free service in the event of any vehicle breakdown. Our application shall make all possible efforts to locate and direct the nearest service provider to the user’s location. The application also helps to call the mechanic to the location of the breakdown. This project is built using react, node, Express, and MySQL. It is also using a neomorphic design that is widely trending today and said to be pleasing to the eyes.

## CHAPTER-2

**WORKING ENVIRONMENT**

**Software Requirement:**

* HTML
* CSS
* JavaScript
* React

**Server language:**

* Node
* Express
* MySQL

**Hardware Requirement:**

* Windows 10

**SYSTEM SOFTWARE**

## HTML

**HTML** or **HyperText Markup Language** is the standard markup language used to create web pages.

HTML was created in 1991 by Tim Berners-Lee at CERN in Switzerland. It was designed to allow scientists to display and share their research.

HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like

<html>). HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent empty elements and so are unpaired, for example <img>. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags).

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language.HTML elements form the building blocks of all websites.

HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages. HTML is a descriptive markup language.

## HTML 5

HTML5 will be the new standard for HTML. The previous version of HTML, HTML 4.01, Came in 1999. The web has changed a lot since then. HTML5 is still a work in progress.

However, the major browsers support many of the new HTML5 elements and APIs.HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).WHATWG was working with web forms and applications, and W3C was working with XHTML 2.0. In 2006, they decided to cooperate and create a new version of HTML.

Some rules for HTML5 were established:

* New features should be based on HTML, CSS, DOM, and JavaScript
* Reduce the need for external plug-ins (like Flash)
* Better error handling
* More markup to replace scripting
* HTML5 should be device-independent.
* The development process should be visible to the public

## 2.1.2 CSS (Cascading Style Sheet):

**CSS tutorial** or CSS 3 tutorial provides basic and advanced concepts of CSS technology. Our CSS tutorial is developed for beginners and professionals. The major points of CSS are given below:

1. CSS stands for Cascading Style Sheet.
2. CSS is used to design HTML tags.
3. CSS is a widely used language on the web.
4. HTML, CSS, and JavaScript are used for web designing. It helps web designers to apply style to HTML tags.

**Cascading Style Sheets** (**CSS**) is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and user interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG, and XUL. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation.

CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, and provide more flexibility and control in the specification of presentation characteristics.

CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader), and on Braille-based, tactile devices.

It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed. While the author of a document typically links that document to a CSS file, readers can use a different style sheet, perhaps one on their computer, to override the one the author has specified.

With plain HTML you define the colors and sizes of text and tables throughout your pages. If you want to change a certain element you will therefore have to work your way through the document and change it. With CSS you define the colors and sizes in “styles”. Then as you write your documents you refer to the styles. Therefore: if you change a certain style, it will change the look of your entire site. Another big advantage is that CSS offers much more detailed attributes than plain HTML for defining the look and feel of your site.

## 2.1.3 JAVASCRIPT

**JavaScript** (**JS**) is a dynamic computer programming language. It is most commonly used as part of web browsers, whose implementations allow client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed. It is also being used in server-side network programming (with Node.js), game development and the creation of desktop and mobile applications.

JavaScript is a prototype-based scripting language with dynamic typing and has first-class functions. Its syntax was influenced by C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages. It is a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles.

The application of JavaScript in use outside of web pages—for example, in PDF documents, site-specific browsers, and desktop widgets—is also significant. Newer and faster JavaScript

VMs and platforms built upon them (notably Node.js) have also increased the popularity of JavaScript for server-side web applications. On the client side, JavaScript was traditionally implemented as an interpreted language but the just-in-time compilation is now performed by recent (post-2012) browsers.

JavaScript was formalized in the ECMA Script language standard and is primarily used as part of a web browser (client-side JavaScript). This enables programmatic access to objects within a host environment.

JavaScript is the most popular programming language in the world.

It is the language for HTML, for the Web, for computers, servers, laptops, tablets, smartphones, and more.

You can use JavaScript to:

* Change HTML elements
* Delete HTML elements
* Create new HTML elements
* Copy and clone HTML elements

## 2.1.4 React

React.js is an open-source JavaScript library that is used for building user interfaces specifically for single-page applications. It’s used for handling the view layer for web and mobile apps. React also allows us to create reusable UI components. React was first created by Jordan Walke, a software engineer working for Facebook. React was first deployed on Facebook’s newsfeed in 2011 and on Instagram.com in 2012.

React allows developers to create large web applications that can change data, without reloading the page. The main purpose of React is to be fast, scalable, and simple. It works only on user interfaces in the application. This corresponds to the view in the MVC template. It can be used with a combination of other JavaScript libraries or frameworks, such as Angular JS in MVC.

React JS is also called simply React or React.js.

React.js properties include the following:

* React.js is declarative
* React.js is simple
* React.js is component-based
* React.js supports server-side
* React.js is extensive
* React.js is fast
* React.js is easy to learn

Why React?

Now, the main question that arises in front of us is why one should use React. There are so many open-source platforms for making front-end web application development easier, like Angular. Let us take a quick look at the benefits of React over other competitive technologies or frameworks. With the front-end world-changing on a daily basis, it’s hard to devote time to learning a new framework – especially when that framework could ultimately become a dead end. So, if you're looking for the next best thing but you're feeling a little bit lost in the framework jungle, I suggest checking out React.

*1. Simplicity*

ReactJS is just simpler to grasp right away. The component-based approach, well-defined lifecycle, and use of just plain JavaScript make React very simple to learn, build a professional web (and mobile applications), and support it. React uses a special syntax called JSX which allows you to mix HTML with JavaScript. This is not a requirement; the Developer can still write in plain JavaScript but JSX is much easier to use.

*2. Easy to learn*

Anyone with a basic previous knowledge in programming can easily understand React while Angular and Ember are referred to as ‘Domain-specific Language’, implying that it is difficult to learn them. To react, you just need basic knowledge of CSS and HTML.

*3. Native Approach*

React can be used to create mobile applications (React Native) and react is a diehard fan of reusability, meaning extensive code reusability is supported. So, at the same time, we can make IOS, Android and Web applications.

*4. Data Binding*

React uses one-way data binding and an application architecture called Flux controls the flow of data to components through one control point – the dispatcher. It's easier to debug self-contained components of large ReactJS apps.

*5. Performance*

 React does not offer any concept of a built-in container for dependency. You can use Browserify, Require JS, and EcmaScript 6 modules which we can use via Babel, and ReactJS-di to inject dependencies automatically.

*6. Testability*

ReactJS applications are super easy to test. React views can be treated as functions of the state, so we can manipulate with the state we pass to the ReactJS view and take a look at the output and triggered actions, events, functions, etc.

## 2.1.5 Node

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. In the following "hello world" example, many connections can be handled concurrently. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep. This is in contrast to today's more common concurrency model, in which OS threads are employed. Thread-based networking is relatively inefficient and very difficult to use. Furthermore, users of Node.js are free from worries of dead-locking the process, since there are no locks. Almost no function in Node.js directly performs I/O, so the process never blocks except when the I/O is performed using synchronous methods of the Node.js standard library. Because nothing blocks, scalable systems are very reasonable to develop in Node.js.

Node.js is similar in design to and influenced by, systems like Ruby's Event Machine and Python's Twisted. Node.js takes the event model a bit further. It presents an event loop as a runtime construct instead of as a library. In other systems, there is always a blocking call to start the event loop. Typically, behavior is defined through callbacks at the beginning of a script, and at the end, a server is started through a blocking call like EventMachine::run(). In Node.js, there is no such start-the-event-loop call. Node.js simply enters the event loop after executing the input script. Node.js exits the event loop when there are no more callbacks to perform. This behavior is like browser JavaScript — the event loop is hidden from the user.

HTTP is a first-class citizen in Node.js, designed with streaming and low latency in mind. This makes Node.js well suited for the foundation of a web library or framework.

Node.js being designed without threads doesn't mean you can't take advantage of multiple cores in your environment. Child processes can be spawned by using our child\_process.fork() API, and are designed to be easy to communicate with. Built upon that same interface is the cluster module, which allows you to share sockets between processes to enable load balancing over your cores.

## 2.1.6 Express

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications

## 2.1.7 MySQL

The database has become an integral part of almost every human's life. Without it, many things we do would become very tedious, perhaps impossible tasks. Banks, universities, and libraries are three examples of organizations that depend heavily on some sort of database system. On the Internet, search engines, online shopping, and even the website naming convention would be impossible without the use of a database. A database that is implemented and interfaced on a computer is often termed a database server. One of the fastest SQL (Structured Query Language) database servers currently on the market is the MySQL server, developed by T.c.X. DataKonsultAB. MySQL, available for download at [www.mysql.com,](http://www.mysql.com/) offers the database programmer an array of options and capabilities rarely seen in other database servers. MySQL is free of charge for those wishing to use it for private and commercial use. Those wishing to develop applications specifically using MySQL should consult MySQL's licensing section, as there is charge for licensing the server.

### These capabilities range across a number of topics, including the following:

1. Ability to handle an unlimited number of simultaneous users.
2. Capacity to handle 50,000,000+ records.
3. Very fast command execution, perhaps the fastest to be found on the market.
4. Easy and efficient user privilege system.

However, perhaps the most interesting characteristic of all is the fact that it's free. That's right,

T.c.X offers MySQL as a free product to the general public.

### Reasons to Use MySQL

1. **Scalability and Flexibility**

The MySQL database server provides the ultimate in scalability, sporting the capacity to handle deeply embedded applications with a footprint of only 1MB to running massive data warehouses holding terabytes of information. Platform flexibility is a stalwart feature of MySQL with all flavors of Linux, UNIX, and Windows being supported.

### High Performance

A unique storage-engine architecture allows database professionals to configure the MySQL database server specifically for particular applications, with the end result being amazing performance results.

**c) High Availability**

Rock-solid reliability and constant availability are hallmarks of MySQL, with customers relying on MySQL to guarantee around-the-clock uptime. MySQL offers a variety of high- availability options from high-speed master/slave replication configurations, to specialized Cluster servers offering instant failover, to third party vendors offering unique high-availability solutions for the MySQL database server.

### Robust Transactional Support

MySQL offers one of the most powerful transactional database engines on the market. Features include complete ACID (atomic, consistent, isolated, durable) transaction support, unlimited row-level locking, distributed transaction capability, and multi-version transaction support where readers never block writers and vice-versa.

### Web and Data Warehouse Strengths

MySQL is the de-facto standard for high-traffic web sites because of its high-performance query engine, tremendously fast data inserts capability, and strong support for specialized web functions like fast full text searches.

### Strong Data Protection

Because guarding the data assets of corporations is the number one job of database professionals, MySQL offers exceptional security features that ensure absolute data protection. In terms of database authentication, MySQL provides powerful mechanisms for ensuring only authorized users have entry to the database server, with the ability to block users down to the client machine level being possible.

### Management Ease

MySQL offers exceptional quick-start capability with the average time from software download to installation completion being less than fifteen minutes. This rule holds true whether the platform is Microsoft Windows, Linux, Macintosh, or UNIX.

SQL functions are implemented using a highly optimized class library and should be as fast as possible. Usually there is no memory allocation at all after query initialization.

## CHAPTER-3

**SYSTEM ANALYSIS**

* 1. **FEASIBILITY STUDY**

The Feasibility study is a test of system proposal according to its workability, impact on the organization, ability to meet user needs and effective use of resources. The object of Feasibility study is not to solve the problem, but to acquire the sense of its scope. During the study the problem definition is crystalized and aspects of the problem to be included in the system are determined, consequently costs and benefits are estimated with greater detail at this stage. The result of Feasibility study is the system formal proposal. This is simply a form of documenting or detailing the nature and scope of proposed solutions. The proposal summarizes what is known and what is going to be done. Three key considerations involved in Feasibility analyses,

* + 1. Economic Feasibility
    2. Technical Feasibility
    3. Behavioral Feasibility

## Economic Feasibility

Economic analysis is the most frequently used method for comparing the cost with the benefit or income that is expected from developed system. In the existing system, many people are involved in this process but in the proposed system, number of persons involved be reduced drastically. So, the proposed system is Economic.

In the existing system, storage of all these records should be arranged and security should be provided for the records. In the proposed system, separate security management is not needed since the application provides security and maintenance is simply and hardly needs one or two persons to operate the system.

## Technical Feasibility

The feasibility center on the existing computer system (application, hardware) and to what extent can extend it can support the proposed addition. In the proposal system data can be easily stored and managed in database management system application. The results for various queries can be generated easily. Therefore, the system is technically feasible.

## Behavioral Feasibility

People are inherently resistant to change and computer has known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have towards the development of computerized system. In the existing system more manpower is required and time factor is more but in the proposed system, power and time factors are reduced. So, the remaining numbers are engaged with some other important works.

## EXISTING SYSTEM

The assistance provided to the travellers is highly limited to the types of service that are being provided. Assistance through the helpline is highly prone to unavailability that makes the traveller’s experience worse. The available services to the travellers are not accessible to the travellers in terms of locality.

## DRAWBACKS OF THE EXISTING SYSTEM

1. It is not a suitable application for emergency needs.
2. It is not feasible. Because if the user is in an unknown location first they find their location using one app and searching a nearby location in another app
3. Requires an active internet connection.
4. System will provide inaccurate results if data is not entered properly.

## PROPOSED SYSTEM

The assistance provided to the travellers are in wide range where they can enjoy all in one manner. The services provided are made available with the information of the service provider with which the traveller can have access. The access to and the presence of service are made to known to the travellers with google map navigation system.

## Benefits of Proposed System

1. Secure registration of users and mechanics.
2. Easy access to the data.
3. The new system is more user-friendly reliable and flexible.
4. Reduce manual work.
5. Search mechanics based on different locations.

## Scope of the project

1. On-Road Vehicle Breakdown Assistance is going to be a good solution for the people who seek help in remote locations with mechanical issues in their vehicles.
2. Admin is allowed to access and manage mechanic details.
3. This online mechanic locator reduces work and can easily find the mechanics from various locations.
4. Users of the On-Road Vehicle Breakdown Assistance will be the registered public and they will be getting connected with the particular mechanic through the trustworthy On-Road Vehicle Breakdown Assistance system.
5. Because only the legally licensed and approved mechanics are enlisted in the [On-Road Vehicle Breakdown Assistance Project](https://codeshoppy.com/shop/product/on-road-vehicle-breakdown-assistance-app/).

## CHAPTER-4 SYSTEM DESIGN

* 1. **DATA FLOW DIAGRAM**

The DFD takes an input-process-output view of a system i.e. data objects flow into the software, are transformed by processing elements, and resultant data objects flow out of the software. Data objects represented by labeled arrows and transformation are represented by circles also called as bubbles. DFD is presented in a hierarchical fashion i.e. the first data flow model represents the system as a whole. Subsequent DFD refine the context diagram (level 0 DFD), providing increasing details with each subsequent level.

The DFD enables the software engineer to develop models of the information domain & functional domain at the same time. As the DFD is refined into greater levels of details, the analyst performs an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of the data as it moves through the process that embody the applications.

A context-level DFD for the system the primary external entities produce information for use by the system and consume information generated by the system. The labeled arrow represents data objects or object hierarchy.

MECHANIC

Send

Request

Provide

Service

View

Information

ADMIN

View

Details

Provide

Approval

**DATABASE**

Search

mechanic

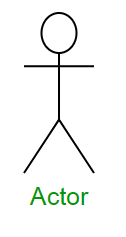
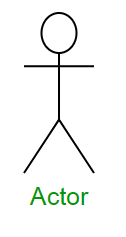
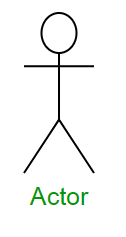
USER

**Figure 4.1 –Data Flow Diagram**

* 1. **USE CASE DIAGRAM**

Use case diagrams model behavior within a system and helps the developers understand of what the user requires. The stick man represents what’s called an actor. Use case diagram can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can’t do. Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

Registration



Login

Approval

Mechanic

User

Post Details

Send Request

View Request

**Figure 4.2 –Use Case Diagram**

* 1. **CLASS DIAGRAM**

Class is nothing but a structure that contains both variables and methods. The Class Diagram shows a set of classes, interfaces, and collaborations and their relating ships. This is the most common diagram in modeling the object-oriented systems and is used to give a static view of a system. It shows the dependency between the classes that can be used in our system.

The interactions between the modules or classes of our projects are shown below. Each block contains Class Name, Variables, and Methods.

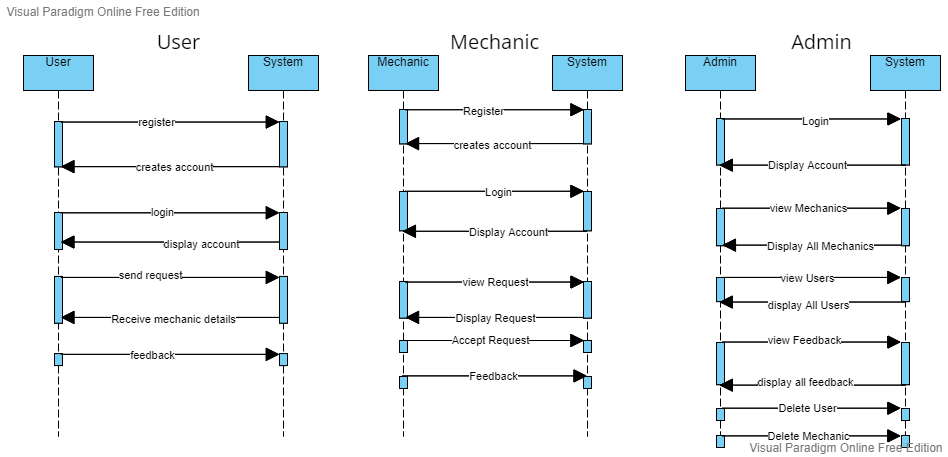
|  |  |  |
| --- | --- | --- |
| **ADMIN** |  | **MECHANIC** |
| + login ()  + View Mechanics ()  + View user ()  + Delete user()  + Delete Mechanic()  + View Feed Back ()  + logout () | + Register ()  + Login ()  + View request ()  + View feedback ()  + logout () |
|  |

|  |
| --- |
| **USER** |
| + register ()  + login ()  +Search Machine ()  + Send Request ()  + Post Feed Back ()  + logout () |

## Figure 4.3 – Class Diagram

* 1. **SEQUENCE DIAGRAM**

A sequence diagram shows an interaction arranged in a time sequence. It shows objects participating in interaction by their lifeline by the message they exchange arranged in time sequence. The vertical dimension represents time and the horizontal dimension represents an object

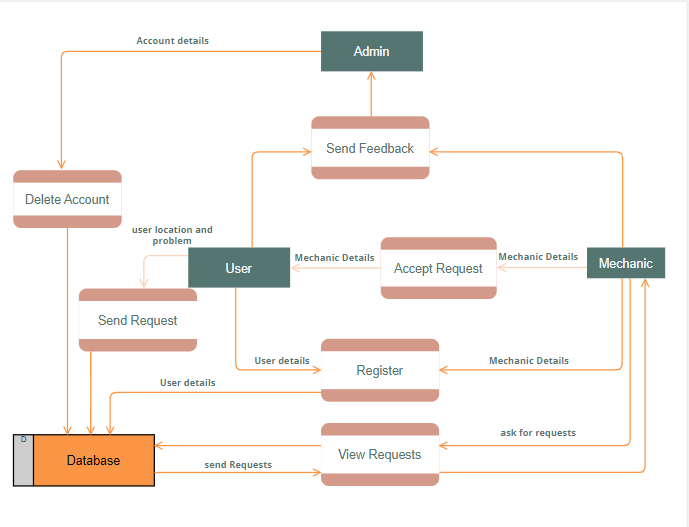
****

**Figure 4.4 – Sequence Diagram**

* 1. **ARCHITECTURE DIAGRAM**

An architecture diagram is a graphical representation of a set of concepts, that are part of the architecture, including their principles, elements, and components.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.



## Figure 4.5 – Architecture Diagram

* 1. **DATABASE DESIGN**

ADMIN:

Table 4.6 -Admin Table

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **TYPE** | **NULL/NOT NULL** | **KEY** |
| Login | INT (11) | NOT NULL | PRIMARY KEY |
| View Mechanics | VARCHAR (30) | NOT NULL |  |
| Provide approval | VARCHAR (30) | NOT NULL |  |
| View User And Their Feedb0ack | VARCHAR (30) | NOT NULL |  |

MECHANIC:

Table 4.7 -Mechanic Table

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **TYPE** | **NULL/NOT NULL** | **KEY** |
| Register | INT (11) | NOT NULL | PRIMARY KEY |
| Login | VARCHAR (50) | NOT NULL |  |
| Post Details | VARCHAR (50) | NOT NULL |  |
| View Request | VARCHAR (50) | NOT NULL |  |
| View Feedback | VARCHAR(50) | NOT NULL |  |

USER:

Table 4.8 -User Table

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **TYPE** | **NULL/NOT NULL** | **KEY** |
| Register | INT (11) | NOT NULL | PRIMARY KEY |
| Login | VARCHAR(50) | NOT NULL |  |
| View Details | VARCHAR(50) | NOT NULL |  |
| Send Request | VARCHAR(50) | NOT NULL |  |
| Post Feedback | VARCHAR(50) | NOT NULL |  |

**CHAPTER-5 PROJECT DESCRIPTION**

* 1. **OBJECTIVE**
* The main objective is to provide a better service and make the process easy and finally appoint a mechanic quickly.
* The proposed system is accessed by three entities namely, Admin, Mechanic, and User.
* A mechanic can perform a task such as viewing requests received from users and can also send feedback to the admin.
* Users can send a request and can appoint a mechanic on the respective date-time.

## MODULE DESCRIPTION MODULES

* **ADMIN**
  + - * Login
      * View mechanics
      * View User
      * View Feedback

## DESCRIPTION:

1. **Login:**

Admin needs to log in with valid login credentials.

1. **View Mechanics:**

Admin can view the registered mechanics with their details and has the access to allow or block a mechanic.

1. **View Users:**

Admin can view all the registered user details.

1. **View Feedback:**

Admin can view all the feedback given by the user and mechanic.

* **Mechanic:** 
  + - * + Register
        + Login
        + View request
        + View feedback

## DESCRIPTION:

## Register:

## Mechanics can register with all their information.

## Login:

## Registered mechanics will be provided access to log in only if the Admin will allow or block it.

## View Request:

## Mechanics can view the request which is sent by the user.

## Feedback:

## Mechanics can provide their feedback.

## USER

* + - Register
    - Login
    - View details
    - Search mechanics
    - Send request
    - Post feedback

## DESCRIPTION:

1. **Register:**

Users can register with all their details.

1. **Login:**

Registered users can log in with their credentials.

1. **Send Request:**

On selection of the mechanics, the user can send the request to the respective mechanic.

1. **Feedback:**

User can give their feedback accordingly.

## IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence to the new system to the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, and an evaluation of change over methods. Apart from planning major tasks of preparing the implementation are the education and training of users. The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system. In-network backup system no, additional resources are needed.

Implementation is the final and the most important phase. The most critical stage in achieving a successful new system is giving the users confidence that the new system will work and be effective. The system can be implemented only after thorough testing is done and if it is found to be working according to the specification.

**CHAPTER -6**

**SYSTEM TESTING**

**6.1 TESTING DEFINITION**

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is Quality assurance.

Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is done. Using detailed testing strategies, a test plan is carried out on each module. The various tests performed in the **“Network Backup System”** are Unit testing, Integration Testing, and User Acceptance testing.

## 6.2 TESTING OBJECTIVE

* + - To find errors in the developed software.
    - To check the working of the function is according to the specification. Their behavior and performance required are fulfilled.
    - To check the reliability and quality of the software.

## 6.3 TYPES OF TESTING

## Unit Testing

The software units in a system are modules and routines that are assembled and integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches the type and size supported by java. The various controls are tested to ensure that each performs its action as required.

## Integration Testing

Data can be lost across any interface, one module can harm another, sub functions when combined, may not produce the desired major functions. Integration testing is a systematic testing to discover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here the server module and client module options are integrated and tested. This testing provides the assurance that the application is well integrated functional unit with smooth transition of data.

## User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the system users at time of developing and making changes whenever required.

## CHAPTER-7 CONCLUSION

* 1. **SUMMARY**

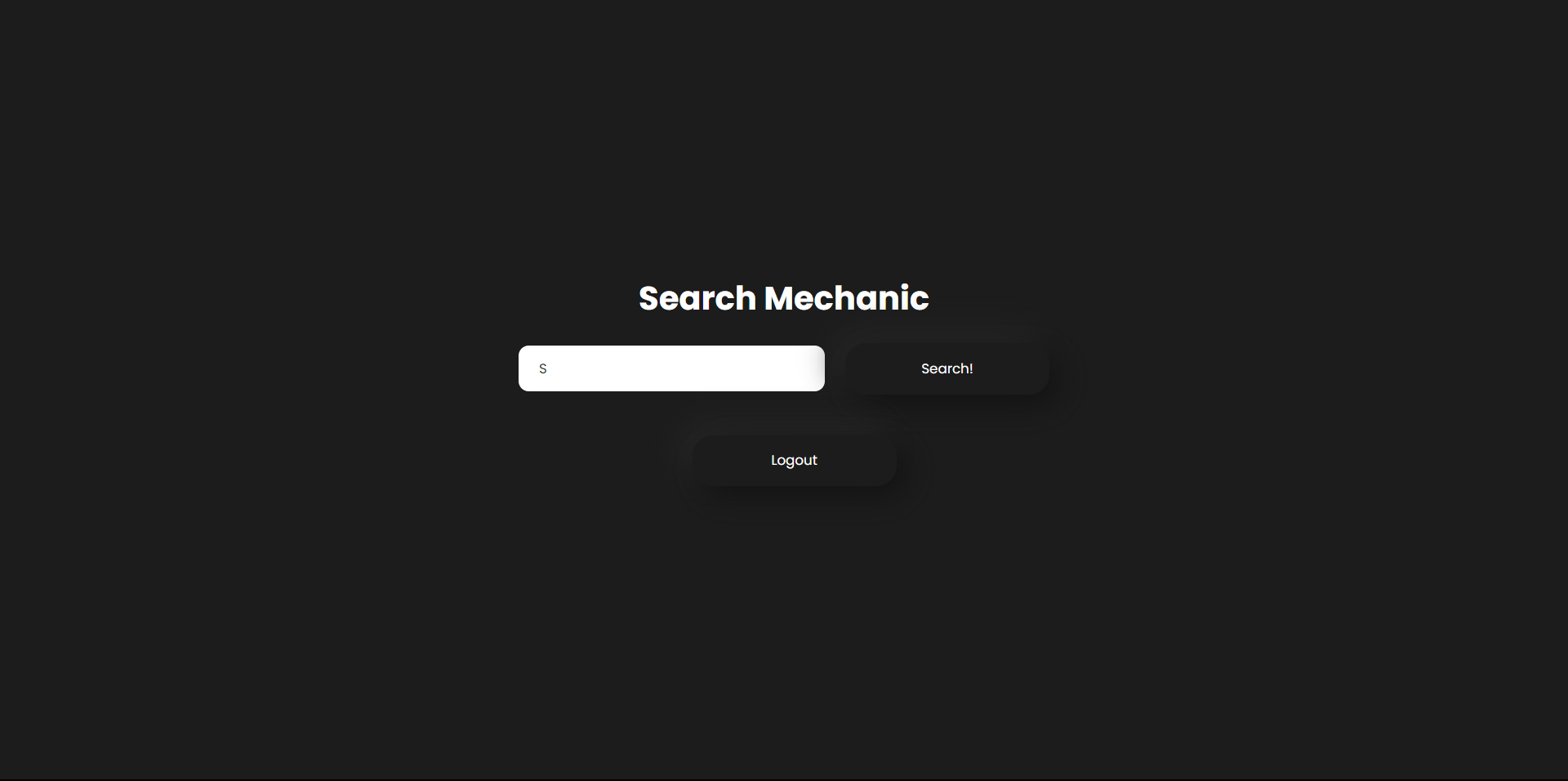
In this Road Vehicle Breakdown Assistance Finder management paper, we presented the design and implementation of an android application called Road assistance system, with which mobile users can get travel-related service information they need anytime and anywhere. The system provides information queries of the Fuel stations, Hospitals, Service station details, and the important services for the travelers like Flat tire service provider details and tow service provider details based on the user’s location.

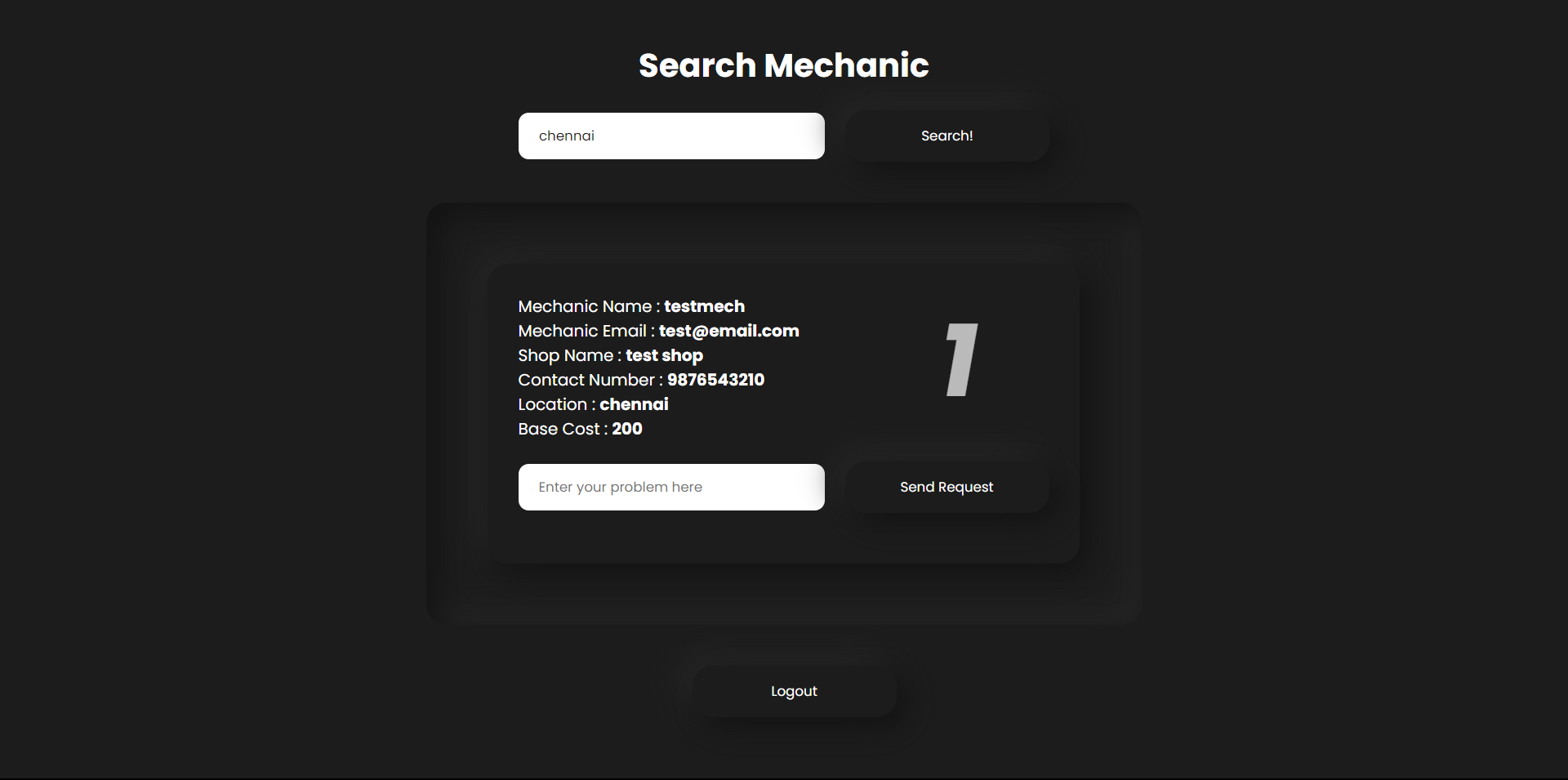
The system is a combination of smartphone and web services and will help tour and life for the user. Tow service details can be accessed from the application, which is stored in the server as part of the broader roadside assistance service. Positioning support (GPS), highlights the user’s current position on the map. The built application successfully provides ease of access (one-touch access) for locating required services.

## FUTURE ENHANCEMENTS

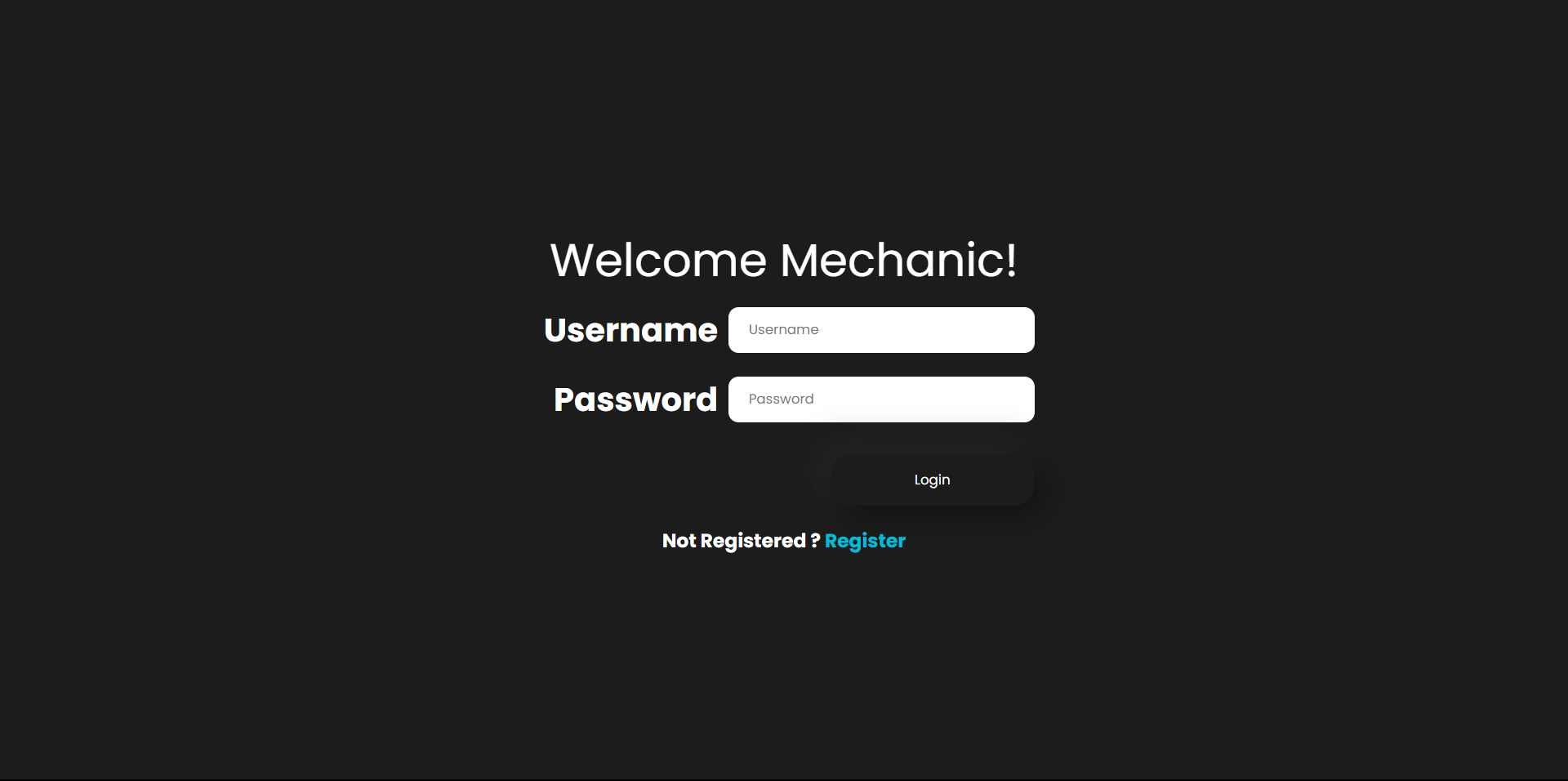
* The world of motoring is changing rapidly, all the time. From electric cars to connected cars to driverless cars, it’s a lot to keep up with.
* But that’s what the RAC has always done, constantly striving to utilize the latest technology to give our members the best breakdown cover possible.
* Pro-active and predictive services are the latest exciting examples that one day soon will be as common as changing a flat tire.

## CHAPTER-8 APPENDIX

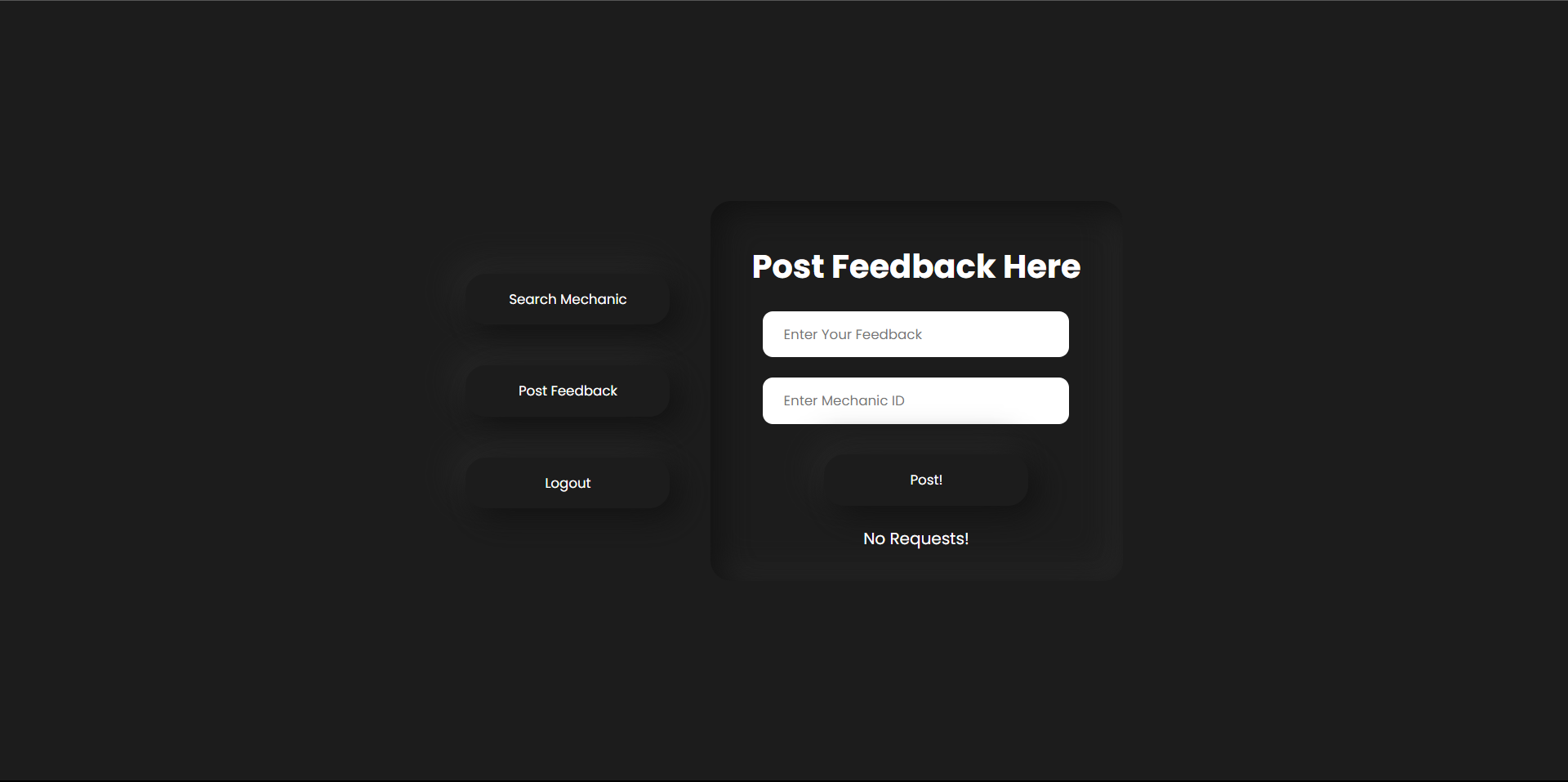
* 1. **SCREENSHOTS**
     1. Find Mechanic



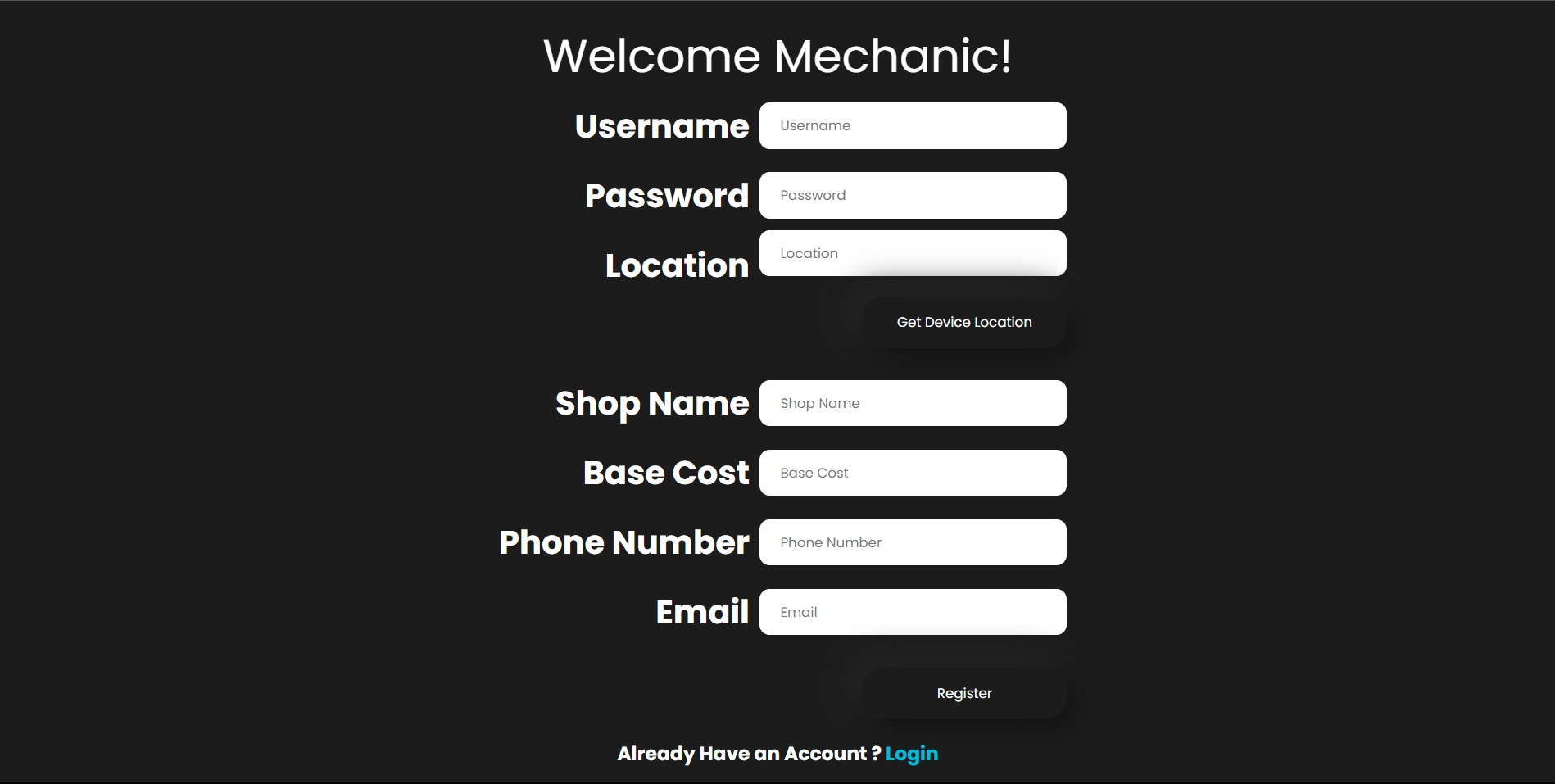
* + 1. Mechanic Login Portal



* + 1. Customer Features



* + 1. Mechanic Registration



* 1. **SAMPLE CODING**

**import React, { useState } from "react";**

**import Axios from "axios";**

**import "./App.css";**

**import { BrowserRouter as Router, Switch, Route, Link } from "react-router-dom";**

**function Register({ user }) {**

**const [Username, setUsername] = useState("");**

**const [Password, setPassword] = useState("");**

**const [Location, setLocation] = useState("");**

**const [Shop, setShop] = useState("");**

**const [Phone, setPhone] = useState(0);**

**const [Cost, setCost] = useState(0);**

**const [Email, setEmail] = useState("");**

**const [VechileType, setVechileType] = useState("");**

**const [VechileModel, setVechileModel] = useState("");**

**const [Problem, setProblem] = useState("");**

**console.log(user);**

**const addUser = () => {**

**Axios.post("http://localhost:3001/addUser", {**

**Username,**

**Password,**

**Location,**

**Phone,**

**Email,**

**VechileType,**

**VechileModel,**

**Problem,**

**}).then(() => {**

**alert("User Added Successfully");**

**window.location.href = "/LoginUser";**

**});**

**};**

**return (**

**<div className="Login flex col">**

**{user === "user" ? <h1>Welcome User!</h1> : <h1>Welcome Mechanic!</h1>}**

**{user === "user" ? (**

**<>**

**<div className="flex col login-container">**

**<div className="input-grp flex row">**

**<h1>Username</h1>**

**<input**

**onChange={(e) => {**

**setUsername(e.target.value);**

**}}**

**type="text"**

**placeholder="Username"**

**/>**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Password </h1>**

**<input onChange={(e) => {**

**setPassword(e.target.value);**

**}} type="password" placeholder="Password" />**

**</div>**

**<div**

**className="input-grp flex row location"**

**>**

**<h1>Location </h1>**

**<div className="flex col right">**

**<input onChange={(e) => {**

**setLocation(e.target.value);**

**}} className="location" id="location" type="text" placeholder="Location" />**

**<button id="locationbutton" onClick={**

**() => {**

**if (navigator.geolocation) {**

**navigator.geolocation.getCurrentPosition(showPosition);**

**}**

**function showPosition(position) {**

**document.getElementById("location").value = "Latitude: " + position.coords.latitude + " Longitude: " + position.coords.longitude;**

**setLocation("Latitude: " + position.coords.latitude + " Longitude: " + position.coords.longitude)**

**}**

**}**

**}>Get Device Location</button>**

**</div>**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Phone Number </h1>**

**<input onChange={(e) => {**

**setPhone(e.target.value);**

**}} type="phone" placeholder="Phone Number" />**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Email </h1>**

**<input onChange={(e) => {**

**setEmail(e.target.value);**

**}} type="email" placeholder="Email" />**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Vehicle Type </h1>**

**<input onChange={(e) => {**

**setVechileType(e.target.value);**

**}} type="text" placeholder="Vehicle Type" />**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Vehicle Model </h1>**

**<input onChange={(e) => {**

**setVechileModel(e.target.value);**

**}} type="text" placeholder="Vehicle Model" />**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Problem Description </h1>**

**<input onChange={(e) => {**

**setProblem(e.target.value);**

**}} type="text" placeholder="Optional" />**

**</div>**

**<button**

**onClick={() => {**

**addUser();**

**}}**

**>**

**Register**

**</button>**

**</div>**

**<h3>**

**Already Have an Account ? <Link to="/LoginUser">Login</Link>**

**</h3>**

**</>**

**) : (**

**<>**

**<div className="flex col login-container">**

**<div className="input-grp flex row">**

**<h1>Username</h1>**

**<input**

**onChange={(e) => {**

**setUsername(e.target.value);**

**}}**

**type="text"**

**placeholder="Username"**

**/>**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Password </h1>**

**<input onChange={(e) => {**

**setPassword(e.target.value);**

**}} type="password" placeholder="Password" />**

**</div>**

**<div**

**className="input-grp flex row location"**

**>**

**<h1>Location </h1>**

**<div className="flex col right">**

**<input onChange={(e) => {**

**setLocation(e.target.value);**

**}} className="location" id="location" type="text" placeholder="Location" />**

**<button id="locationbutton" onClick={**

**() => {**

**if (navigator.geolocation) {**

**navigator.geolocation.getCurrentPosition(showPosition);**

**}**

**function showPosition(position) {**

**document.getElementById("location").value = "Latitude: " + position.coords.latitude + " Longitude: " + position.coords.longitude;**

**setLocation("Latitude: " + position.coords.latitude + " Longitude: " + position.coords.longitude)**

**}**

**}**

**}>Get Device Location</button>**

**</div>**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Shop Name</h1>**

**<input onChange={(e) => {**

**setShop(e.target.value);**

**}} type="text" placeholder="Shop Name" />**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Base Cost </h1>**

**<input onChange={(e) => {**

**setCost(e.target.value);**

**}} type="text" placeholder="Base Cost" />**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Phone Number </h1>**

**<input onChange={(e) => {**

**setPhone(e.target.value);**

**}} type="phone" placeholder="Phone Number" />**

**</div>**

**<div**

**className="input-grp flex row"**

**>**

**<h1>Email </h1>**

**<input onChange={(e) => {**

**setEmail(e.target.value);**

**}} type="email" placeholder="Email" />**

**</div>**

**<button**

**onClick={() => {**

**console.log(**

**Username +**

**" " +**

**Password +**

**" " +**

**Email +**

**" " +**

**Cost +**

**" " +**

**Phone +**

**" " +**

**Shop +**

**" " +**

**Location**

**);**

**Axios.post("http://localhost:3001/addMech", {**

**Username,**

**Password,**

**Location,**

**Shop,**

**Cost,**

**Phone,**

**Email,**

**}).then((res) => {**

**console.log(res.data);**

**alert("Mechanic Added Successfully");**

**window.location.href = "/LoginMech";**

**});**

**}}**

**>**

**Register**

**</button>**

**</div>**

**<h3>**

**Already Have an Account ? <Link to="/LoginMech">Login</Link>**

**</h3>**

**</>**

**)}**

**</div>**

**);**

**}**

**export default Register;import React, { useEffect } from "react";**

**import logo from "./asset/img/logo.jpeg";**

**import "./App.css";**

**function Intro() {**

**useEffect(() => {**

**setTimeout(() => {**

**var element = document.getElementById("intro-logo");**

**element.classList.add("loaded");**

**}, 1000);**

**}, []);**

**return (**

**<div className="flex intro">**

**<img className="" id="intro-logo" src={logo} alt="" />**

**</div>**

**);**

**}**

**export default Intro;  
  
  
import React, { useEffect, useState } from "react";**

**import Axios from "axios";**

**import { BrowserRouter as Router, Switch, Route, Link } from "react-router-dom";**

**import deleteIcon from "./asset/img/delete.png";**

**import "./App.css";**

**function Admin() {**

**const [Login, setLogin] = useState(false);**

**const [Username, setUsername] = useState("");**

**const [Password, setPassword] = useState("");**

**const [viewMech, setViewMech] = useState(false);**

**const [viewUser, setViewUser] = useState(false);**

**const [MechData, setMechData] = useState([]);**

**const [UserData, setUserData] = useState([]);**

**useEffect(() => {**

**Axios.get("http://localhost:3001/getMechanics")**

**.then((response) => {**

**setMechData(response.data);**

**})**

**.then(() => {**

**setViewMech(!viewMech);**

**});**

**Axios.get("http://localhost:3001/getUsers")**

**.then((response) => {**

**setUserData(response.data);**

**})**

**.then(() => {**

**setViewUser(!viewUser);**

**});**

**}, []);**

**const deleteMechanic = (id) => {**

**Axios.delete(`http://localhost:3001/deleteMechanic/${id}`).then(() => {**

**alert("Mechanic Deleted Successfully");**

**window.location.reload();**

**});**

**};**

**const deleteUser = (id) => {**

**Axios.delete(`http://localhost:3001/deleteUser/${id}`).then(() => {**

**alert("User Deleted Successfully");**

**window.location.reload();**

**});**

**};**

**return (**

**<div className="Admin flex col">**

**<h1>Welcome Admin!</h1>**

**{!Login ? (**

**<div className="flex col login-container">**

**<div className="input-grp flex row">**

**<h1>Username</h1>**

**<input**

**onChange={(e) => {**

**setUsername(e.target.value);**

**}}**

**type="text"**

**placeholder="Username"**

**/>**

**</div>**

**<div**

**onChange={(e) => {**

**setPassword(e.target.value);**

**}}**

**className="input-grp flex row"**

**>**

**<h1>Password </h1>**

**<input type="password" placeholder="Password" />**

**</div>**

**<button**

**onClick={() => {**

**if (Username == "Admin" && Password == "Admin@123") {**

**setLogin(true);**

**} else {**

**alert("Invalid Username or Password");**

**}**

**}}**

**>**

**Login**

**</button>**

**</div>**

**) : (**

**<>**

**<div className="list-wrapper">**

**<button><Link to="/">**

**Logout**

**</Link></button>**

**</div>**

**<div className="list-wrapper">**

**<div className="view-mechanics">**

**<h1>Mechanics Data</h1>**

**{viewMech ? (**

**<>**

**{MechData.map((data, index) => (**

**<div className="mechanic-card flex row" key={index}>**

**<div className="left flex col">**

**<div className="content">**

**Username : {data.username}**

**</div>**

**<div className="content">Email : {data.email}</div>**

**<div className="content">**

**Location : {data.location}**

**</div>**

**</div>**

**<img**

**onClick={() => {**

**deleteMechanic(data.mechid);**

**}}**

**src={deleteIcon}**

**alt=""**

**/>**

**<div className="left flex col">**

**<div className="content">**

**Contact Number : {data.phone\_num}**

**</div>**

**<div className="content">**

**Shop Name : {data.shop\_name}**

**</div>**

**<div className="content">**

**Base Cost : {data.base\_cost}**

**</div>**

**</div>**

**</div>**

**))}**

**</>**

**) : (**

**<>No Mechanics Found!</>**

**)}**

**</div>**

**<div className="view-mechanics view-user">**

**<h1>Users Data</h1>**

**{viewUser ? (**

**<>**

**{UserData.map((data, index) => (**

**<div className="mechanic-card flex row" key={index}>**

**<div className="left flex col">**

**<div className="content">**

**Username : {data.username}**

**</div>**

**<div className="content">Email : {data.email}</div>**

**<div className="content">**

**Location : {data.location}**

**</div>**

**<div className="content">Problem : {data.problem}</div>**

**</div>**

**<img**

**onClick={() => {**

**deleteUser(data.userid);**

**}}**

**src={deleteIcon}**

**alt=""**

**/>**

**<div className="left flex col">**

**<div className="content">**

**Contact Number : {data.phone\_num}**

**</div>**

**<div className="content">**

**Vehicle Type : {data.vehicle\_type}**

**</div>**

**<div className="content">**

**Vehicle Model : {data.vehicle\_model}**

**</div>**

**</div>**

**</div>**

**))}**

**</>**

**) : (**

**<>No Users Found!</>**

**)}**

**</div>**

**</div>**

**</>**

**)}**

**</div>**

**);**

**}**

**export default Admin;  
  
  
  
import React, { useContext, useState } from "react";**

**import { BrowserRouter as Router, Switch, Route, Link } from "react-router-dom";**

**import "./App.css";**

**import { UserContext } from "./App";**

**import Axios from "axios";**

**function Login({ user }) {**

**const [Username, setUsername] = useState("");**

**const [Password, setPassword] = useState("");**

**const { CurrentUser, setCurrentUser } = useContext(UserContext);**

**const { CurrentUserType, setCurrentUserType } = useContext(UserContext);**

**const [IsValidUser, setIsValidUser] = useState(false);**

**const checkUser = () => {**

**Axios.get(**

**user === "user"**

**? `http://localhost:3001/user`**

**: `http://localhost:3001/mech`,**

**{**

**params: {**

**username: Username,**

**password: Password,**

**},**

**}**

**).then((response) => {**

**if (response.data.length > 0) {**

**console.log(response.data);**

**setIsValidUser(true);**

**setCurrentUser(response.data[0]);**

**setCurrentUserType(user);**

**} else {**

**alert("Invalid Username or Password");**

**}**

**});**

**};**

**return (**

**<div className="Login flex col">**

**{user === "user" ? <h1>Welcome User!</h1> : <h1>Welcome Mechanic!</h1>}**

**<div className="flex col login-container">**

**<div className="input-grp flex row">**

**<h1>Username</h1>**

**<input**

**onChange={(e) => {**

**setUsername(e.target.value);**

**}}**

**type="text"**

**placeholder="Username"**

**/>**

**</div>**

**<div**

**onChange={(e) => {**

**setPassword(e.target.value);**

**}}**

**className="input-grp flex row"**

**>**

**<h1>Password </h1>**

**<input type="password" placeholder="Password" />**

**</div>**

**<button onClick={() => checkUser()}>Login</button>**

**{IsValidUser && user === "user" ? (**

**<Link to={"/UserDashboard"}>**

**<button>Proceed</button>**

**</Link>**

**) : (**

**""**

**)}**

**{IsValidUser && user === "mech" ? (**

**<Link to={"/MechDashboard"}>**

**<button>Proceed</button>**

**</Link>**

**) : (**

**" "**

**)}**

**</div>**

**<h3>**

**Not Registered ?{" "}**

**<Link to={user === "user" ? `/RegisterUser` : `/RegisterMech`}>**

**Register**

**</Link>**

**</h3>**

**</div>**

**);**

**}**

**export default Login;**

**CHAPTER-9 BIBLIOGRAPHY AND REFERENCE**

* 1. **BIBLIOGRAPHY**
* https://www.tutorialspoint.com/index.htm
* https://www.javatpoint.com
* https://www.w3schools.com
* https://html.com

## REFERENCES

* Anon., 2019. *You tube.* [Online] Available at: https://www.youtube.com/watch?v=E1eqRNTZqDM&t=551s [Accessed 15 02 2020].
* Anon., 2020. *Git Hub.* [Online] Available at: https://github.com/ [Accessed 20 02 2020].
* firebase, 2020. *Firebase Documentation.* [Online] Available at: https://firebase.google.com/docs/auth/android/start [Accessed 03 02 2020].
* Florian, e., 2017. *Google Patent.* [Online] Available at: https://patents.google.com/patent/US20190171758A1/en [Accessed 17 January 2020].
* Masahiko, e., 2000. *Google Patents.* [Online] Available at: https://patents.google.com/patent/US6972669B2/en [Accessed 20 October 2019].
* Monica, 2018. A Car Breakdown Service Station Locator System. *INTERNATIONAL JOURNAL OF ADVANCE SCIENTIFIC RESEARCH,* 3(4), pp. 13-16.
* Morales, O., 2016. *Google Patent.* [Online] Available at: https://patents.google.com/patent/US10234299B2/en [Accessed 17 January 2020].
* Reichardt, e., 2002. *Car Talk 2000.* [Online] Available at: https://ieeexplore.ieee.org/abstract/document/1188007 [Accessed 17 December 2019 ].
* Sophie, N., 2001. *Google patent.* [Online] Available at: https://patents.google.com/patent/US6973387B2/en [Accessed 5 January 2020].
* The Interaction Design Foundation. (2020). *Prototyping: Learn Eight Common Methods and* (Anon., 2020)*Best Practices*. [online] Available at: https://www.interaction-design.org/literature/article/prototyping-learn-eight-common-methods-and-best-practices [Accessed 20 Jan. 2020].