Table of Contents

Table of Figures

List of Tables

Online Mechanic Locator

Chapter 1: INTRODUCTION

We all love our Cars and we want them to be in the same new condition as they came from factory. This sounds good but to make it happen, we require a regular servicing of our Car. You may have encountered a situation wherein you are in need of a mechanic to fix your car but you don't want to give it to a random local mechanic whom you don't know and you're skeptical about his skills or the amount of money he is charging.

Once your car goes out of warranty you must have faced the hassle of going to your Dealer's Garage for servicing your Car where they will definitely charge you more money than what they used to charge you before when the car was still in warranty. You also have to wait for days or sometimes weeks to get your car back, because they have a large pool of car lined in queue to be repaired and you don't know how long it will take for your car's turn to come for repair.

The system which we are building will make Car Owner's life easy, making car repair and maintenance affordable, convenient, and transparent. Car Owners can find best offers and deals from various different Garages. The system allows Mechanics to showcase their skills and will get online presence where they can solve problem faced by Car Owners and can reach to a larger audience. Hence, making a good sum of money through our system. Our system also allows Garages to increase their sales by showing their services & offers to Car Owners and motor parts listing to Mechanics, directly through the system.

1.1 Background:

Dealing with car problems is not only a time-consuming and frustrating experience, it is often expensive. Car owners often lack visibility regarding the quality of mechanics or fairness of the price. At the same time, the mechanics who put in the hard work fixing cars make very little money at repair shops (less than 20% of what consumers pay), and rarely get the recognition they deserve.

The current situation is depicted in Figure 1 below:-

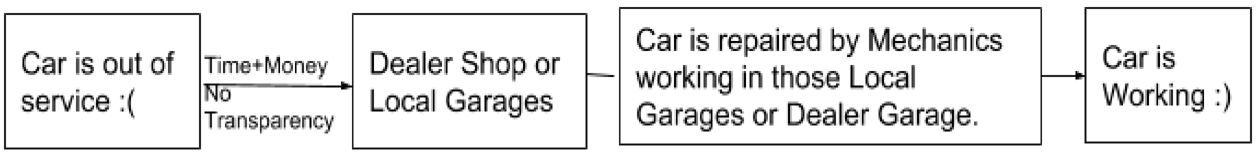

Car is out of service :(
Time+Money
No Transparency
Dealer Shop or Local Garages
Car is repaired by Mechanics working in those Local Garages or Dealer Garage.
Car is Working :)

Figure 1.1 Problem Situation

As you can see in the diagram that when car owners go to the repair shop (Dealer Shop or Local Garage), they don't have any metrics to compare the prices for what they've been charged for. It is also a very time-consuming process and you also have to trust your dealer that whether he is installing genuine motor parts or not. Because it has happened in many cases that dealer install faulty auto parts in your car which will work for some months and then it will again fail so you have to come again for repairing and again the whole cycle goes on and on.
Mechanics who are skilled at their work don't get the recognition they deserve. They often work under some garage where they are paid less.
I believe that the current auto repair system is broken and needs a new system where Car Owners will have better experience and it will empower mechanics to live a better life. It will help garages increase their sales by selling their products and services to the appropriate customer through our system. The system will connect Mechanics, Car Owners and Garages at a single place where they can work together to solve the problem mentioned above.

1.2 Objective:

- To help Car Owners find best deals and services from Mechanics and Garages.
- To empower mechanics to live a better life by making his online presence where he can reach to a larger audience and get paid well by fixing the car issue.
- To make Car Owner's life easy by removing all the hassles of going to the service center for repairing the car.
- To boost the sales of products and services of Garages through the system.
- The car owner has the option to select services provided by the specific Garage or specific Mechanic.

1.3 Purpose, Scope and Applicability

1.3.1 Purpose:

The title itself gives an indirect clue about this project. "Online Mechanic Finder" you can find Mechanics Online to repair your Cars. The project is being executed because there is a need of a system which will allow the users to freely communicate and ask for certain services from the Mechanics.
The Sole purpose is to provide Users a medium through which they can easily, efficiently, securely and comfortably seek help for Car services in a very interactive and friendly User-Interface environment.

Firstly, the Car Owner creates his/her account in the application where he/she will be asked for details about their Cars. When the user experiences any problem in their car, instead of going

to a Garage the user can find Mechanics on our system and can chat with him. The Owner can also broadcast his car problem so that it will reach to every mechanic in the system and whoever is ready to solve the Owner's Problem can directly chat with Owner and thus can tell his proposed price quote to the Owner. In this way Owner will receive many price quote by many different mechanics and Owner can choose between mechanics based on their reviews, ratings, past work history and his proposed price quote.
Secondly, the Mechanic himself has a profile on the system through which they can showcase their skills and can reach to a massive audience, providing their services and thus can earn a good amount of money. They can also book slots in Garage for repair space or can see the garages product (i.e. Auto parts) listing and can contact or chat with Garage for bulk orders at discounted price. Mechanic can chat with Owners through Online Chatting feature provided in the system which is secured and responsive.
Through this system Garage can increase their sales by showcasing their products(Auto parts) and services(towing, repair slots, etc) to Mechanics and by attracting Car Owners through lucrative offers like Free Car wash, Free Inspection and Diagnostic, etc.

1.3.2 Scope:

- Storing information of Owners, Mechanics & Garages.
- Check validity of information provided by Owners and Mechanics.
- Giving the ability to chat within the system through an interactive and user friendly chatting portal.
- Storing the reviews and ratings provided by car owners.
- Storing the information of garages like offers and products.
- Giving the ability to mechanics to earn through the system by connecting to Owners and Garages.
  Storing owners problem in database

1.3.3 APPLICABILITY:

- The System help to connect Car Owners, Mechanics and Garages, where these entities can work together for the benefit of each other.
- This application registers Customer, Mechanics and Garages profiles.
- It has a Mechanic Profile & Garage Profile listing which makes it convenient for the Owner to select and appoint a Mechanic or select a Garage service.
- Different type of Vehicle services are provided to the user.
- Owner can book a slot for his/her vehicle in a specific garage for repair.
- Mechanics can contact garage for purchasing automobile parts or book slot for repairing the car.
- Garage has the ability to share his product listing and services to mechanics and owners respectively. Thus, increasing the sales.

1.4 Achievements

This project has helped me in understanding deeply about the concept of how HTTP works and why it is an important protocol. I have learned about Web Services and how they are responsible for sharing data from remote server to application. ReST API is what makes it possible to transfer data to Android Application in JSON format which can be then used to display the data on your application.

I have learned about various tools and technology that can be used to make Android Application in much faster and efficient way. I have learned about various JavaScript and web frameworks which are amazing and can be used to make cross platform application. I have also learned about various design tool to design mockup and prototype at early stage which helps to show the flow of the application.

When I was researching about database i came across Firebase which is NoSQL database and the marketing done by Google Team has initially made me believe that it is the best option available to make application. Though, i agree that firebase is amazing and it has many great features but that is just one side of story. The other side is SQL databases, it's been around from over 30 years and the features it offers is also amazing. This confusion has given boost to my curiosity to know the difference between the two and I have documented differences in chapter 2.

I have also developed a good understanding of how to collect important and relevant data from google. This project has helped me in developing my research ability.

I understood the importance of Software Engineering and why it is important to properly plan your project before implementing it. It will save you a bunch of time and conceptual model helps in communicating with other stakeholders of the project.

I believe that there is many more thing to learn and explore that will help me in making this system more efficient and I will surely learn about them when i am at implementation stage.

1.5 Organization of Report

Chapter 1 provides a brief introduction about the project, its objectives and goals accomplished. It also focuses on the scope and purpose of the project and how it can be applied in the real world.

2nd chapter focuses on the technologies that can be used for making this system. It shows that how mobile application can be developed in much faster and efficient way by making use of cross platform application building tool like Cordova. Later in this chapter we see various other JavaScript libraries and framework that can be used for making UI components that are fully responsive. It also explain in brief the most heated debate among the developer community i.e. SQL vs NoSQL.

3rd chapter describes the problem definition of the traditional Auto Repair System. It also outlines the drawbacks of the existing system that lead to requirements to eliminate the problems of the old system. It also covers the requirements of the software and hardware components that I will be using to make this project. Also, the way I have planned to complete this project in stipulated amount of time will be shown with the help of Gantt chart. The Gantt chart describes how I have scheduled my whole project completion.

In the end of this chapter, I have shown the initial conceptual representation of how the data will be stored and what is the relation among different entities with of help of ER diagram.

4th chapter focuses on defining the modules which will be there in the system. It then shows the Schema Design and Constraint which the data has in the system. It will also focus the way data is being used and handled, sent and received, updated and deleted. It will also show a basic user interface that will help to get an idea of how the system will look like to the user. It also discusses security concerns regarding the project and also the testing phases that will help to eliminate errors and mistakes at the initial stages of project implementation that will try to reduce the error in least amount of time and efforts.

The 5th chapter will focus on the implementation of system design. This will include implementation approaches, important snippet of codes, the kind of testing approaches to be taken and their details.

The 6th chapter gives results of the project implementation. It shows the results of what was completed. It will discuss these results in different scenarios and will help to determine the success rate in all types of scenarios. It will also provide a user documentation of the system, which will include all the necessary details required to use the system and get it working immediately.

The 7th chapter will conclude the project, describing the limitations of the system and how they can be eliminated in the future. It will also include ideas in the improvements in of the project which might not be feasible at present, but have a future scope.

Chapter 2: SURVEY OF TECHNOLOGIES

Molding an Idea into Reality requires a good code. Code is that ingredient that helps in shaping your idea and bringing it to reality by helping in developing a good system. For making something which is not present in the market, it requires good and deep understanding of the problem. If the problem is properly understood than we can search for the technologies which can be used to solve that problem in faster and much efficient way.
Below are the survey of some technologies which can be used to make the system.

2.1 Mobile Application Development

2.1.1 Native App

Native application is a software or program which has been developed to perform some specific task on particular environment or platform. Native application built using software development tools (SDK) for a certain software framework, hardware platform or operating system. Like Android app built using Java Development Kit on Java platform, iOS app built using iOS SDK, Swift and Objective C. Similarly, .NET required for Windows platform.

Advantages:

- Graphical Applications, HD games, intensive animation applications might perform well as native app because Native code is still faster than HTML and JavaScript. WebGL standards helps browser and hybrid app for gaming apps to meet performance but still native has edge.
- Native SDKs allows to access device features without dealing with complexity of native plugins and new device features will be available out of the box along with SDKs.
- Not much dependencies on open source libraries and platforms like Cordova and Ionic.

Disadvantages:

- Separate development effort for each platform which increases the development time.
- Each platform code will have its own release cycle and updates which adds to development time and cost.
- Releasing same feature on all platform at same time always challenging because of different code base.

Different skill set required to develop and maintain the same application on each platform which adds to the cost.

2.1.2 Hybrid App

Hybrid apps are native apps only because it can be downloaded from platform's app store like native app. It can get access to all the native platform features. It can have performance close to native app. The major differences are listed below:

- Hybrid apps are built using web technologies like HTML, CSS and JavaScript whereas Native apps built with specific technology and language for specific platform like Java for Android, Swift for iOS.
- Hybrid app runs in webView (A view that displays web pages, uses the same engine of browser but no browser like widgets)
- Native plugins required to access the native features of the platform like camera, mic etc. (Native plugins are like wrapper on top of native libraries or components)
- Hybrid app can be built for any platform from single code base.

Libraries and Frameworks:

Cordova is an open-source mobile development framework. Cordova Plugin helps to access device features. It allows to use standard web technologies for cross-platform development. Applications execute within wrappers targeted to each platform.

Ionic is the app platform for web developers. We can build amazing mobile, web, and desktop apps all with one shared code base and open web standards. It uses Cordova behind the scene.

Other cross platform app development frameworks are listed below:

- Framework7
- Titanium Appcelerator
- OnsenUI
- Xamarin (based on C#)

Advantages:

- Single code base for all platforms means write once and run anywhere but for native app scenario, we need to build and maintain separate app and code for each platform.
- Same development team can deliver app for any platform including website as well because all required is web technologies.
- Hybrid App is based on web technologies, so same app can be run on browser like any other website or can be run as Progressive Web App(PWA).
- Hybrid app can achieve the same hardware-based performance acceleration as native app.
- Hybrid app can have same and consistent user experience across platform regardless of user moves between different devices or browser.

<u>Disadvantage:</u>

- For most applications, performance is same as native app but 3D, HD games, high graphics-oriented apps and other performance centric apps, hybrid approach might not go well.
- Hybrid app can able access all the native device features like touchId, media etc. but dependent on native plugins. Sometime entire new device feature might be not being readily available as native plugin. We can write our own but it adds complexity to the development.

Hybrid app is having dependencies on different libraries and frameworks like Cordova, Ionic which have to be in sync with latest platform version changes and releases.

2.2 Web Application Development

2.2.1 Frontend Development

<u>HTML</u>

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

<u>CSS</u>

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css
file, and reduce complexity and repetition in the structural content.

<u>JavaScript</u>

JavaScript often abbreviated as JS, is a high-level, interpreted Programming language. It is a language which is also characterized as dynamic, weakly typed, prototype-based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

Frameworks and Libraries:-

Bootstrap

Twitter Bootstrap is the most popular front end framework in the recent time. It is sleek, Intuitive, and powerful mobile first front-end framework for faster and easier web development. It uses HTML, CSS and JavaScript. Bootstrap is a free and open-source front-end framework for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions.

AngularJS

- Angular is superheroic JavaScript MVVM (Model-View-ViewModel) framework, founded in 2009, which is awesome for building highly interactive web applications.
- Companies that use Angular : Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather,
  iStockphoto, AWS, Crunchbase.

ReactJS

- ReactJS is a JavaScript library, open sourced by Facebook in 2013, which is great for building
  Huge web applications where data is changeable on a regular basis.
- Companies that use ReactJS: Facebook, Instagram, Netflix, New York Times, Yahoo, Khan Academy, Whatsapp, Codecademy, Dropbox, Airbnb, Asana, Atlassian, Intercom, Microsoft.

Vue.JS

- Vue.js is a JavaScript framework, launched in 2013, which perfectly fits for creating highly adaptable user interfaces and sophisticated Single-page applications.
- Companies that use Vue.js: Xiaomi, Alibaba, WizzAir, EuroNews, Grammarly, Gitlab and Laracasts, Adobe, Behance, Codeship, Reuters.

JavaScript frameworks are developing at an extremely fast pace, meaning that today we have frequently updated versions of Angular, ReactJS and another player on this market—Vue.js. We analyzed the number of open positions worldwide that require a specific knowledge of a certain framework. As a source, we took Indeed.com and got the distribution (in right) according to more than 60,000 job offers.

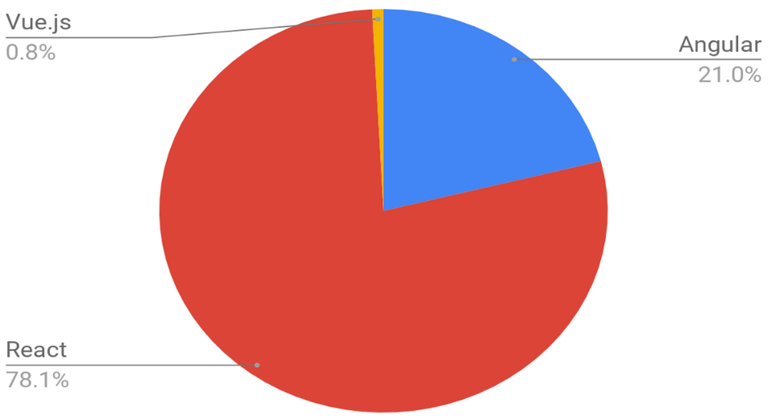


Figure 2.1 JS Framework Comparison

2.2.2 Backend Development

<u>PHP</u> (Hypertext PreProcessor)
It is a widely-used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML.
What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve. The best things in using PHP are that it is extremely simple for a newcomer, but offers many advanced features for a professional programmer.

<u>Features of PHP</u>

It is most popular and frequently used world wide scripting language, the main reason of popularity is; It is open source and very simple.

- Simple
- Faster
- Interpreted
- Open Source
- Case Sensitive
- Simplicity
- Platform Independent
- Security
- Loosely Typed Language

Laravel

Laravel is a free, open-source PHP web framework, created by Taylor Otwell and intended for the development of web applications following the model–view–controller (MVC)

Architectural pattern and based on Symfony.

Python

Python is a beautiful language. It’s easy to learn and fun, and its syntax (the rules) is clear and concise. Python is a popular choice for beginners, yet still powerful enough to back some of the world’s most popular products and applications from companies like NASA, Google, IBM, Cisco, Microsoft, Industrial Light & Magic among others. One area where Python shines is web development. Python offers many frameworks from which to choose from including bottle.py, Flask, CherryPy, Pyramid, Django and web2py. These frameworks have been used to power some of the world’s most popular sites such as Spotify, Mozilla, Reddit, the Washington Post and Yelp.

Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

- Ridiculously fast - Django was designed to help developers take applications from concept to completion as quickly as possible.
- Reassuringly secure - Django takes security seriously and helps developers avoid many common security mistakes.
  Exceedingly scalable - Some of the busiest sites on the Web leverage Django’s ability to quickly and flexibly scale.

2.3 Database

When it comes to choosing a database, one of the biggest decisions is picking a relational (SQL) or non-relational (NoSQL) data structure. While both are viable options, there are certain key differences between the two that must be kept in mind when making a decision.
Here, I break down the most important distinctions and discuss two of the key players in the relational vs non-relational debate: MySQL and MongoDB.

The Big Picture Differences

The Language

Think of a town - we'll call it Town A - where everyone speaks the same language. All of the businesses are built around it, every form of communication uses it - in short, it's the only way that the residents understand and interact with the world around them. Changing that language in one place would be confusing and disruptive for everyone.

Now, think of another town, Town B, where every home can speak a different language. Everyone interacts with the world differently, and there's no "universal" understanding or set organization. If one home is different, it doesn't affect anyone else at all.

This helps illustrate one of the fundamental differences between SQL relational and NoSQL non-relational databases, and this distinction has big implications. Let's explain:

SQL databases use structured query language (SQL) for defining and manipulating data. On one hand, this is extremely powerful: SQL is one of the most versatile and widely-used options available, making it a safe choice and especially great for complex queries. On the other hand, it can be restrictive. SQL requires that you use predefined schemas to determine the structure of your data before you work with it. In addition, all of your data must follow the same structure. This can require significant up-front preparation, and, as with Town A, it can mean that a change in the structure would be both difficult and disruptive to your whole system.

A NoSQL database, on the other hand, has dynamic schema for unstructured data, and data is stored in many ways: it can be column-oriented, document-oriented, graph-based or organized as a Key-Value store. This flexibility means that:

- You can create documents without having to first define their structure
- Each document can have its own unique structure
- The syntax can vary from database to database, and
- You can add fields as you go.

The Scalability: In most situations, SQL databases are vertically scalable, which means that you can increase the load on a single server by increasing things like CPU, RAM or SSD. NoSQL databases, on the other hand, are horizontally scalable. This means that you handle more traffic by sharing, or adding more servers in your NoSQL database. It's like adding more floors to the same building versus adding more buildings to the neighborhood. The latter can ultimately become larger and more powerful, making NoSQL databases the preferred choice for large or ever-changing data sets.

The Structure: SQL databases are table-based, while NoSQL databases are either document-based, key-value pairs, graph databases or wide-column stores. This makes relational SQL databases a better option for applications that require multi-row transactions - such as an accounting system - or for legacy systems that were built for a relational structure.

Some examples of SQL databases include MySQL, Oracle, PostgreSQL, and Microsoft SQL Server. NoSQL database examples include MongoDB, FireBase, BigTable, Redis, RavenDB Cassandra, HBase, Neo4j and CouchDB.

SQL vs NoSQL: MySQL vs MongoDB

Now that we've established the key structural differences between SQL and NoSQL databases, let's delve into the key functional differences between the two, looking specifically at MySQL and FireBase as examples.

MySQL: The SQL Relational Database

The following are some MySQL benefits and strengths:

- **Maturity**: MySQL is an extremely established database, meaning that there's a huge community, extensive testing and quite a bit of stability.
- **Compatibility:** MySQL is available for all major platforms, including Linux, Windows, Mac, BSD and Solaris. It also has connectors to languages like Node.js, Ruby, C#, C++, Java, Perl, Python and PHP, meaning that it's not limited to SQL query language.
- **Cost-effective:** The database is open source and free.
- **Replicable:** The MySQL database can be replicated across multiple nodes, meaning that the workload can be reduced and the scalability and availability of the application can be increased.
- **Sharding:** While sharding cannot be done on most SQL databases, it can be done on MySQL servers. This is both cost-effective and good for business.

MongoDB: The NoSQL Non-Relational Database

The following are some of MongoDB benefits and strengths:

- **Dynamic schema:** As mentioned, this gives you flexibility to change your data schema without modifying any of your existing data.
- **Scalability:** MongoDB is horizontally scalable, which helps reduce the workload and scale your business with ease.
- **Manageability:** The database doesn't require a database administrator. Since it is fairly user-friendly in this way, it can be used by both developers and administrators.
- **Speed:** It's high-performing for simple queries.
- **Flexibility:** You can add new columns or fields on MongoDB without affecting existing rows or application performance.

Side-by-Side Comparison of MySQL and MongoDB:

	MySQL	MongoDB
Written in	C++, C	C++, C and JavaScript
Type	RDBMS	Document-oriented
Main points	• Table • Row • Column	• Collection • Document • Field
Schemas	Strict	Dynamic
Scaling	Vertically	Horizontally

Key features	● Full-text searching and indexing ● Integrated replication support ● Triggers ● SubSELECTs ● Query caching ● SSL support ● Unicode support ● Different storage engines with various performance characteristics	● Auto-sharding ● Native replication ● In-memory speed ● Embedded data models support ● Comprehensive secondary indexes ● Rich query language support ● Various storage engines support
Best used for	● Data structure fits for tables and rows ● Strong dependence on multi-row transactions ● Frequent updates and modifications of large volume of records ● Relatively small datasets	● High write loads ● Unstable schema ● Your DB is set to grow big ● Data is location based ● HA (high availability) in unstable environment is required ● No database administrators (DBAs)
Examples	NASA, US Navy, Bank of Finland, UCR, Walmart, Sony, S2 Security Corporation, Telenor, Italtel, iStock, Uber, Zappos, Booking.com, Twitter, Facebook, others.	Expedia, Bosch, Otto, eBay, Gap, Forbes, Foursquare, Adobe, Intuit, Metlife, BuzzFeed, Crittercism, CitiGroup, the City of Chicago, others.

Table 3.1 MySQL VS MongoDB

Pros and Cons Table:

MySQL pros	MongoDB pros
● Atomic transactions support ● JOIN support ● Mature solution ● Privilege and password security system	● Document validation ● Integrated storage engines ● Shortened time between primary failure and recovery
MySQL cons	**MongoDB cons**
● Tough scaling ● Stability concerns ● Isn't community-driven development	● Not the best option for apps with complex transactions ● Not a snap-in replacement for legacy solutions ● Young solution

Table 3.2 Pros and Cons of SQL and MongoDB

Chapter 3: REQUIREMENT AND ANALYSIS

3.1 Problem Definition

When the car is in warrantee, the servicing is usually provided by Manufacturer's Official Repair Centre but when the car is out of warranty and we want to service our cars, we all have faced the problems like waiting in a long queue for your turn to come for repairing, there‘s lot of hassle in dealing with the Dealer who you know is charging you more money than what he is supposed to be charging. You will eventually end up paying more than the actual work done on the car. In today's world where time plays an important role is everyone's life, it is very hard to go to dealer shop for car servicing because of your busy schedule. You also don't know anything about your mechanics like his experience, his ratings and reviews because there is no such system to give rating and reviews. The other cost-effective way is to go to a local garage. They will charge you less than the Dealer but still the problem of visiting there by making some time out of your busy schedule still exists. Also the Mechanics working in these types of the Local Garage are making way lower income than what other mechanics of same expertise are getting in Dealers or some Official Manufacturer's service center. India is a place where people with high skills in these types of work (e.g. servicing) are getting paid lower than industry standards because they don't have any platform to reach the masses and they end up working in some local garages, where they are underpaid.

So I think that the current system for automobile repair is broken and need some change. By using my skills which I have acquired in IT, I want to change this whole system by providing an infrastructure for mechanics to get to the mass audience (here, car owner) also wants to provide a medium for car owner to find and book mechanics.

<u>Existing Applications similar to Online Mechanic Finder:</u>

- YourMechanic
  YourMechanic was founded in 2012 with the goal to make car repair and maintenance, affordable, convenient, and transparent.
- SuperCheapAuto
  SuperCheapAuto is a thriving specialty retail business, specializing in automotive parts and accessories. It is an Australian based car repair company

Though these startups are solving the same problem which we are solving but the way of solving the problem is completely different. They provide mechanics under their brand so mechanic themselves could not create their own brand while using their system. This limitations is solved by our system where mechanics can create a good reputation and their own brand by using our system which let them connect to massive user base. We are also keeping the garages in our system giving them the ability to reach to massive audience and thus increasing their sales. Hence, we are creating an online platform where Owners can be benefitted by getting a good mechanic to repair his car when he needs, Mechanics can increase their earnings and can also book services and products from garages and lastly garages can increase their sales and create online presence through our system.

3.2 Requirements Specification

The system provides Car Owners to choose a highly skilled good mechanic who can either take your car to any nearby garage or will take it to his own place, will repair it and then will drop it to your place or you can drop your car to the mechanics place and he'll use his skills to work perfectly on your car and eventually fix it.

Because this system is changing the whole automobile repair system, it cannot happen overnight. Hence, we are connecting the current local garage to our system where we are focusing on increasing the sales and services provided by those garages. For e.g. Mechanic can purchase the parts provided by these Local Garages. Car Owners can avail some special offer like free Car Inspection, free Car Wash, Towing Services, Repair Services provided by these Local Garages. This will help boost the sales of these Garages and so will also make sure that our system is not completely demolishing the current system and therefore, our system will grow and change the current workflow of how these Local Garage operates.

We are on a mission to change the current Automobile Repair System by interconnecting Car Owner, Garages and Mechanics. This system has the potential to change the current automobile repair industry.

3.3 Planning and Scheduling

Software development model:

This project will be developed using the Iterative Model of Software Development process. An **Iterative Life Cycle Model** does not start with a full specification of requirements. In this model, the development begins by specifying and implementing just part of the software, which is then reviewed in order to identify further requirements. Moreover, in iterative model, the iterative process starts with a simple implementation of a small set of the software requirements, which iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed. Each release of Iterative Model is developed in a specific and fixed time period, which is called iteration.
Iterative model can be used in the following scenario:-

- When the requirements of the complete system are clearly defined and understood.

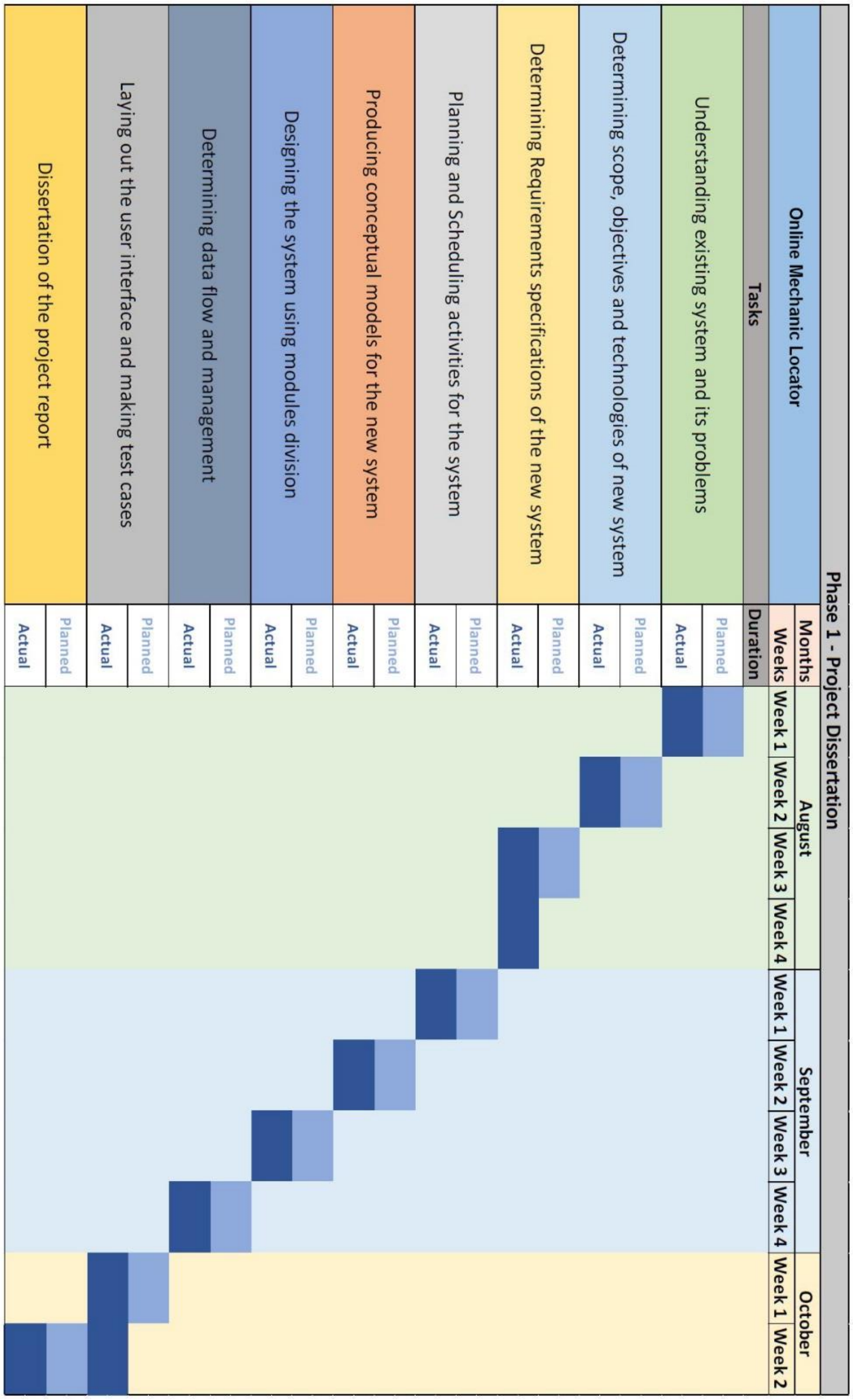
- The major requirements are defined, while some functionalities and requested enhancements evolve with the process of the development process.
- A new technology is being used and is being learnt by the development team, while they are working on the project.
- If there are some high risk features and goals, which might change in the future.
- When the resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.

The above characteristics makes iterative model suitable for developing this System where we want to develop a Minimum Viable Product (MVP) of Online Mechanic Locator and deploy in the market to get reviews and report from the market to incorporate in the system and release the next version with extended features and functionalities.

Process of Iterative model:

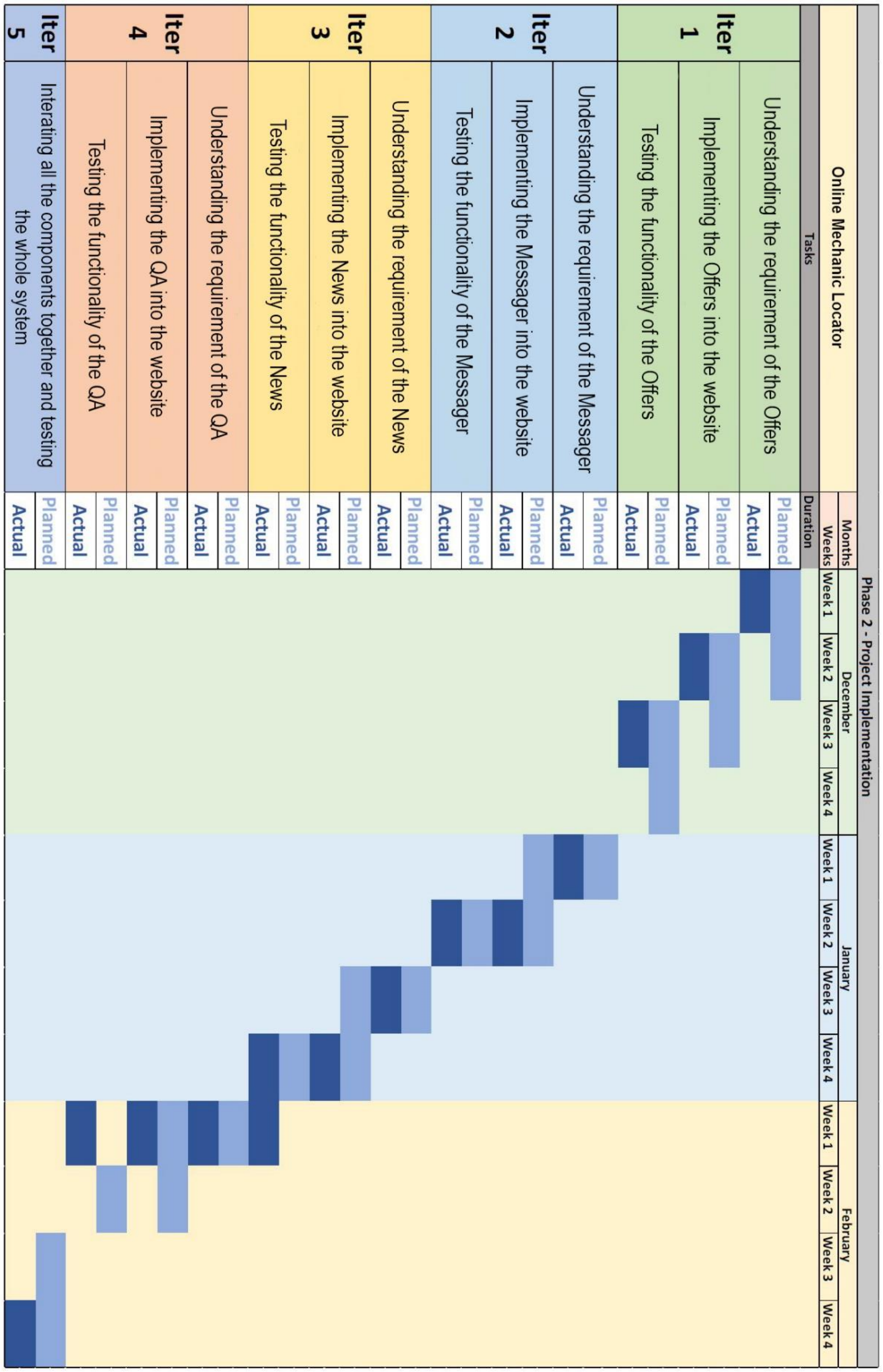
The process of Iterative Model is cyclic, unlike the more traditional models that focus on a rigorous step-by-step process of development. In this process, once the initial planning is complete, a handful of phases are repeated again and again, with the completion of each cycle incrementally improving and iterating on the software.

Gantt chart:



Phase 1 - Project Dissertation											
Online Mechanic Locator	Months	August				September				October	
	Weeks	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2
Tasks	Duration										
Understanding existing system and its problems	Planned	■									
	Actual	■									
Determining scope, objectives and technologies of new system	Planned		■								
	Actual		■								
Determining Requirements specifications of the new system	Planned			■							
	Actual			■	■						
Planning and Scheduling activities for the system	Planned					■					
	Actual					■					
Producing conceptual models for the new system	Planned						■				
	Actual						■				
Designing the system using modules division	Planned							■			
	Actual							■			
Determining data flow and management	Planned								■		
	Actual								■		
Laying out the user interface and making test cases	Planned									■	
	Actual									■	■
Dissertation of the project report	Planned										■
	Actual										■

Figure 3.1 Gantt chart – Phase 1



Online Mechanic Locator			Phase 2 - Project Implementation											
		Months	December				January				February			
	Tasks	Weeks	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4	Week 1	Week 2	Week 3	Week 4
		Duration												
Iter 1	Understanding the requirement of the Offers	Planned	🟦	🟦										
		Actual	🟦											
	Implementing the Offers into the website	Planned		🟦	🟦									
		Actual		🟦										
	Testing the functionality of the Offers	Planned			🟦	🟦								
		Actual			🟦									
Iter 2	Understanding the requirement of the Messager	Planned					🟦							
		Actual					🟦							
	Implementing the Messager into the website	Planned					🟦	🟦						
		Actual						🟦						
	Testing the functionality of the Messager	Planned						🟦						
		Actual						🟦						
Iter 3	Understanding the requirement of the News	Planned							🟦					
		Actual							🟦					
	Implementing the News into the website	Planned							🟦	🟦				
		Actual								🟦				
	Testing the functionality of the News	Planned								🟦				
		Actual								🟦	🟦			
Iter 4	Understanding the requirement of the QA	Planned									🟦			
		Actual									🟦			
	Implementing the QA into the website	Planned									🟦	🟦		
		Actual									🟦			
	Testing the functionality of the QA	Planned										🟦		
		Actual									🟦			
Iter 5	Interating all the components together and testing the whole system	Planned											🟦	🟦
		Actual												🟦

Figure 3.2 Gantt chart – Phase 2

3.4 Software and Hardware Requirements

Software Requirements:

The following are the recommended requirements for the smooth functioning of the entire system:

Software	Requirements
Operating system Platform	**Windows 10**
Browser	**Any of Chrome, Mozilla, Opera etc.**
Development Environment	**Sublime Text 3, Django Server, Cordova or PhoneGap**
Development Tools	**HTML, CSS, Bootstrap, JavaScript Framework, Django**
Database	**MySQL**

Table 3.3 Software Requirement

Hardware Requirements:

The following are the minimum requirements for the smooth functioning of the entire system:

Hardware	Requirements
Processor	**Intel CORE i3 7th Gen**
RAM	**4 GB or Higher**
Hard Disk	**250 Gb or Higher**
Internet	**High-speed Internet connection**

Table 3.4 Hardware Requirement

3.5 Preliminary Product Description

The system is very innovative and has the potential to change how people do servicing of their car. It has the potential to provide employment to many mechanics who have the skills but don't have an exposure to the market.

Following are Preliminary Product Description:

- Car Owners can find and book Mechanics through this system at the comfort of sitting at their home without having to go to garage for repairing their cars.
- Car Owners can also find good Deals and Offers from various different Garage through this system.
- Through this system, Mechanic can earn money by getting connected to huge community of Car Owners where he can provide his services.
- Garages can provide offers and product listing through this system thus increasing their sales.

3.6 Conceptual Models

Class Diagram:

A class diagram is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Purpose of Class Diagrams:

- Shows static structure of classifiers in a system
- Diagram provides basic notation for other structure diagrams prescribed by UML
- Helpful for developers and other team members too
- Business Analysts can use class diagrams to model systems from business perspective.

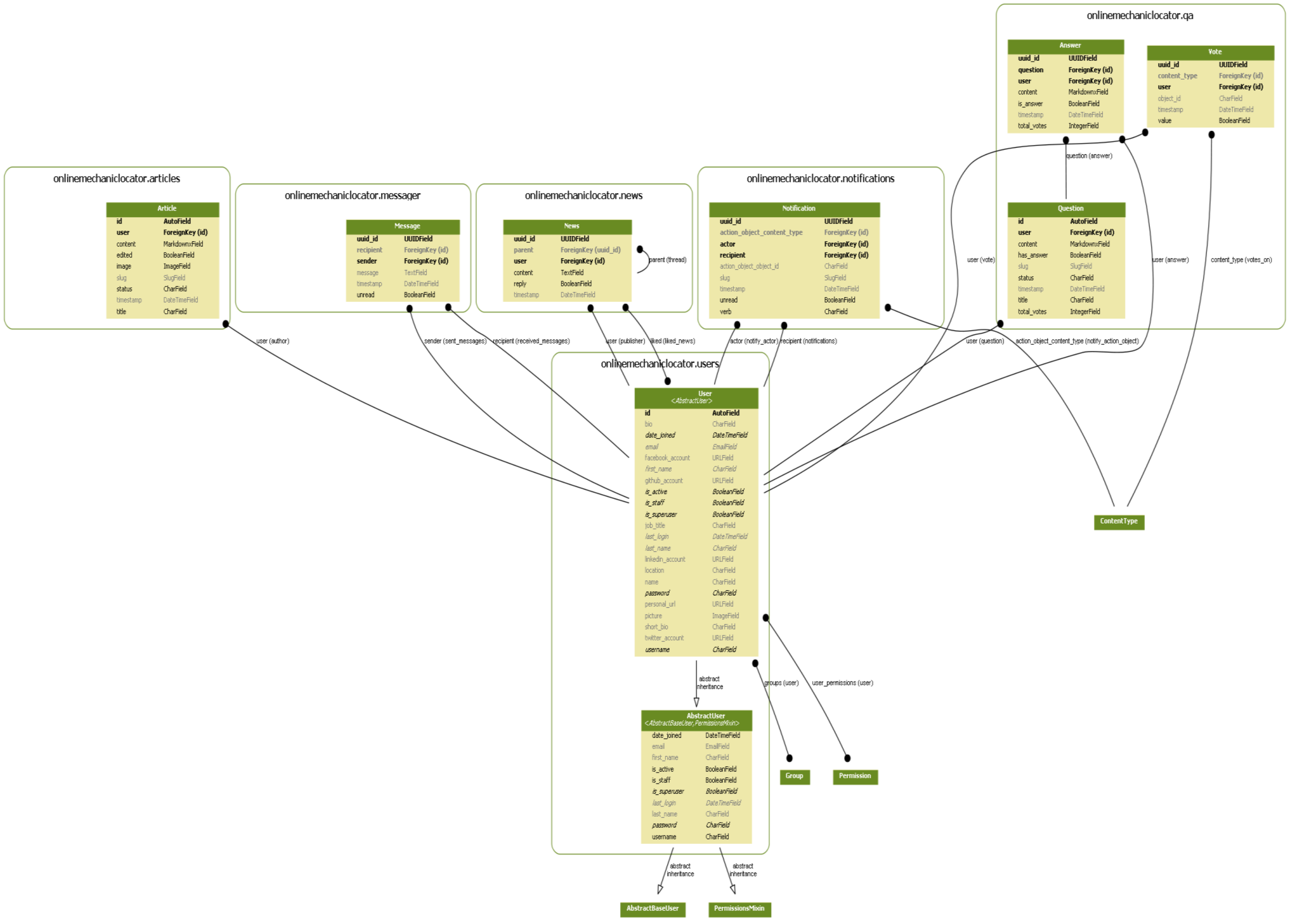


Figure 3.3 Class Diagram

CHAPTER 4: SYSTEM DESIGN

4.1 Basic Modules

There will be three modules in this system i.e. Owner, Mechanic, Garage/
The details of these modules is given below.

1. **Owner**

- Login/Register.
- Edit their Profile.
- Can see and locate Mechanics Profiles, Garages offers and services.
- Can broadcast their car problem, it can be seen by all the Mechanics in the system.
- Can tell problem to a particular Mechanic, it can only be seen by that Mechanic.
- Can chat with Mechanics.
- Can give ratings and reviews to Mechanics and Garages.
- Can report Garages and Mechanics.

2. **Mechanic**

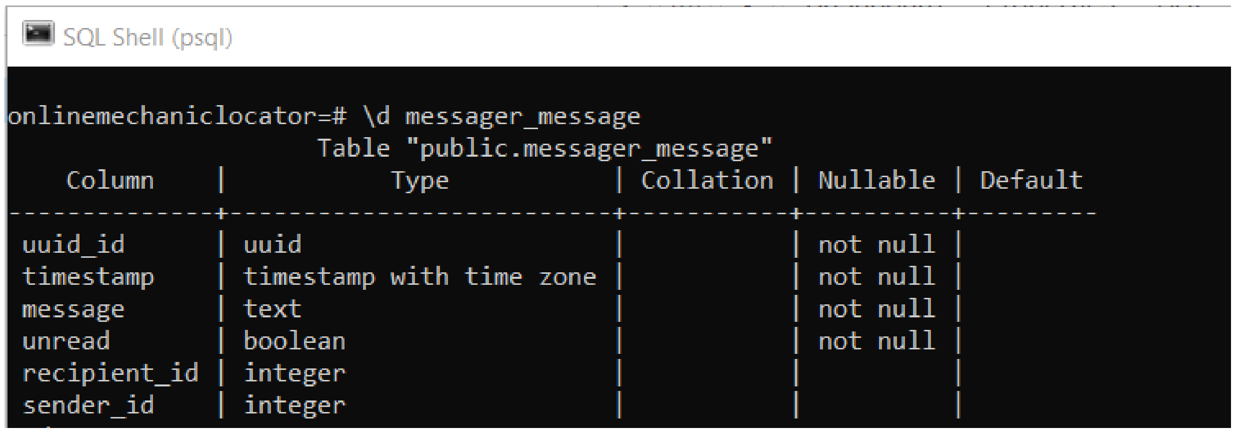
- Login//Register.
- Edit their Profiles.
- Can see the broadcasted problems and confirm or ignore them. If confirmed then can chat with Owner and solve his/her problems.
- Can see his personalized question asked by User and if agrees than can chat with Owner.
- Can see various Garages nearby and see their services like motor parts, slots for repairing is open or not.
- Can chat or call Garage for booking slots or purchasing motor parts.
- Can Report Garages.

3. **Garages**

- Login/Registration.
- Can sell their services (e.g. Car Wash, Free Inspection of Car) to Owners.
- Can sell motor parts and slot booking facility to Mechanics.
- Can chat with Mechanics.
- Can report Mechanics.

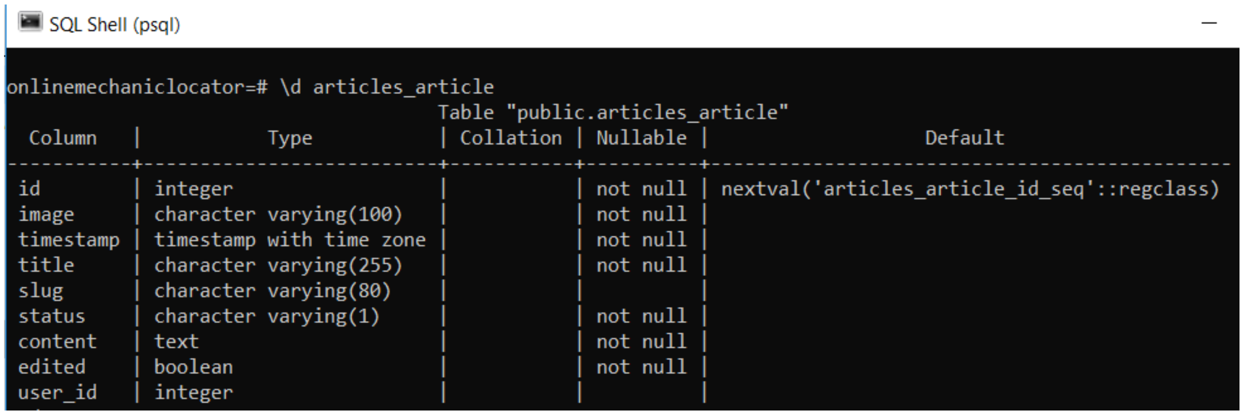
4.2 Data Design

4.2.1 Schema Design



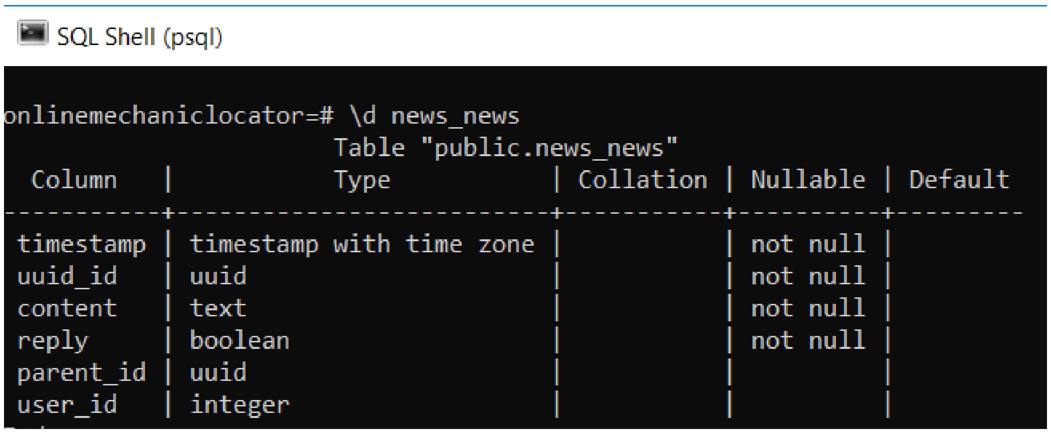
```
onlinemechaniclocator=# \d messager_message
                    Table "public.messager_message"
    Column    |           Type           | Collation | Nullable | Default
--------------+--------------------------+-----------+----------+---------
 uuid_id      | uuid                     |           | not null |
 timestamp    | timestamp with time zone |           | not null |
 message      | text                     |           | not null |
 unread       | boolean                  |           | not null |
 recipient_id | integer                  |           |          |
 sender_id    | integer                  |           |          |
```

Table 4.1 Messager Table



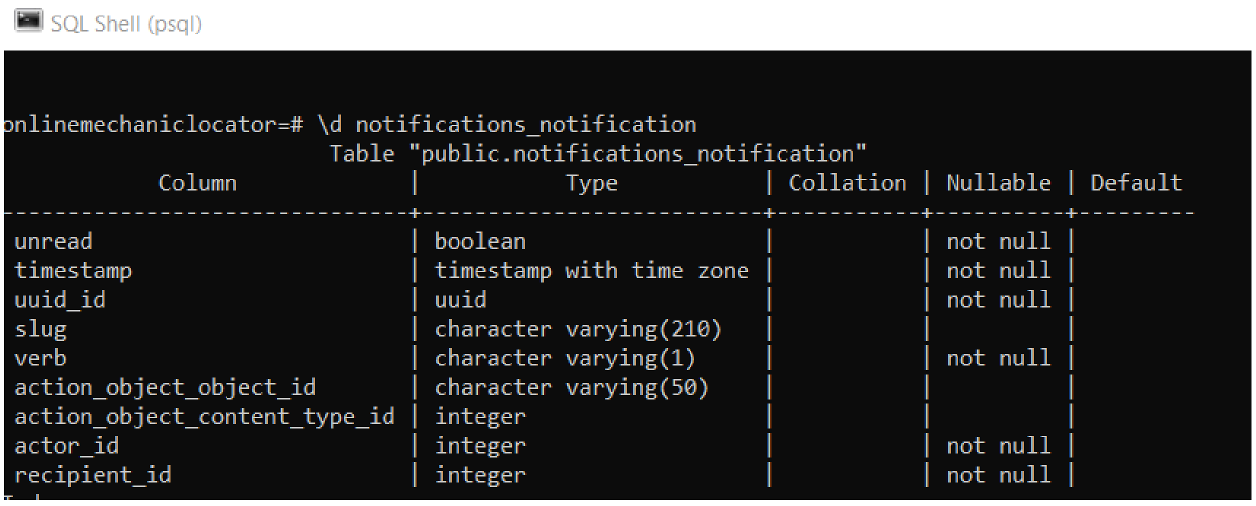
```
onlinemechaniclocator=# \d articles_article
                                      Table "public.articles_article"
  Column   |           Type           | Collation | Nullable |                   Default
-----------+--------------------------+-----------+----------+----------------------------------------------
 id        | integer                  |           | not null | nextval('articles_article_id_seq'::regclass)
 image     | character varying(100)   |           | not null |
 timestamp | timestamp with time zone |           | not null |
 title     | character varying(255)   |           | not null |
 slug      | character varying(80)    |           |          |
 status    | character varying(1)     |           | not null |
 content   | text                     |           | not null |
 edited    | boolean                  |           | not null |
 user_id   | integer                  |           |          |
```

Table 4.2 Article Table



```
onlinemechaniclocator=# \d news_news
                        Table "public.news_news"
  Column   |           Type           | Collation | Nullable | Default
-----------+--------------------------+-----------+----------+---------
 timestamp | timestamp with time zone |           | not null |
 uuid_id   | uuid                     |           | not null |
 content   | text                     |           | not null |
 reply     | boolean                  |           | not null |
 parent_id | uuid                     |           |          |
 user_id   | integer                  |           |          |
```

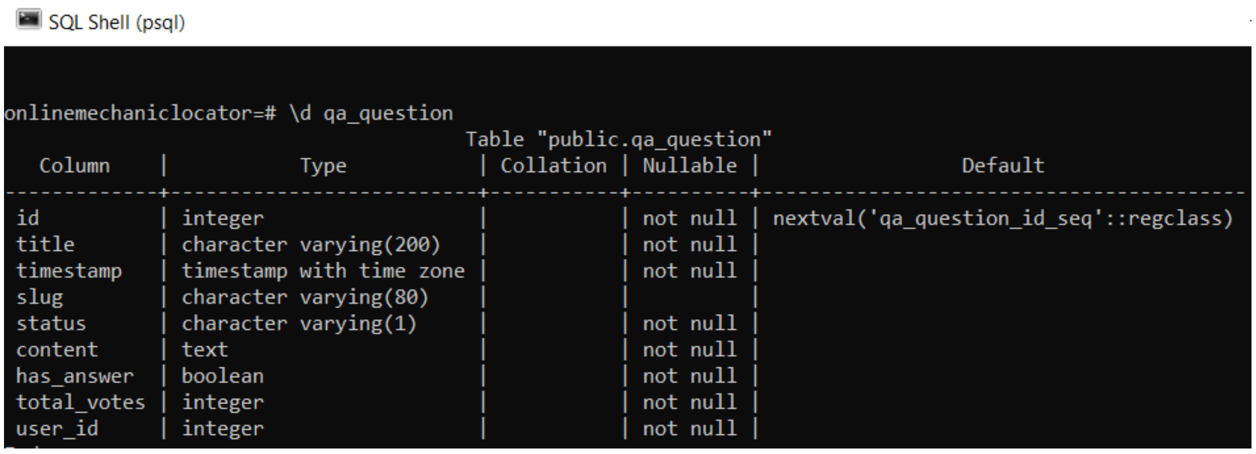
Table 4.3 News Table



SQL Shell (psql)

```
onlinemechaniclocator=# \d notifications_notification
                            Table "public.notifications_notification"
            Column            |           Type           | Collation | Nullable | Default
------------------------------+--------------------------+-----------+----------+---------
 unread                       | boolean                  |           | not null |
 timestamp                    | timestamp with time zone |           | not null |
 uuid_id                      | uuid                     |           | not null |
 slug                         | character varying(210)   |           |          |
 verb                         | character varying(1)     |           | not null |
 action_object_object_id      | character varying(50)    |           |          |
 action_object_content_type_id | integer                 |           |          |
 actor_id                     | integer                  |           | not null |
 recipient_id                 | integer                  |           | not null |
```

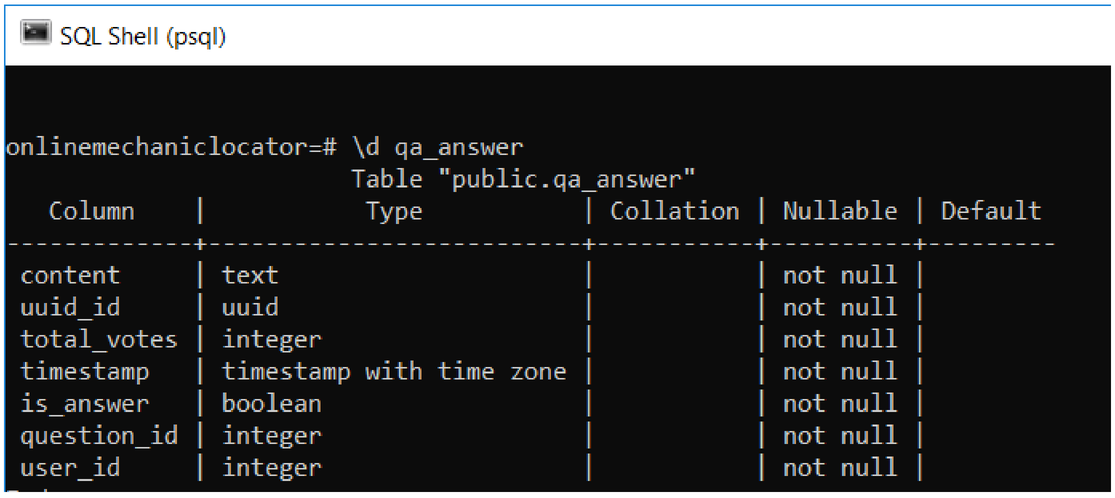
Table 4.4 Notification Table



SQL Shell (psql)

```
onlinemechaniclocator=# \d qa_question
                                       Table "public.qa_question"
   Column    |           Type           | Collation | Nullable |                 Default
-------------+--------------------------+-----------+----------+-----------------------------------------
 id          | integer                  |           | not null | nextval('qa_question_id_seq'::regclass)
 title       | character varying(200)   |           | not null |
 timestamp   | timestamp with time zone |           | not null |
 slug        | character varying(80)    |           |          |
 status      | character varying(1)     |           | not null |
 content     | text                     |           | not null |
 has_answer  | boolean                  |           | not null |
 total_votes | integer                  |           | not null |
 user_id     | integer                  |           | not null |
```

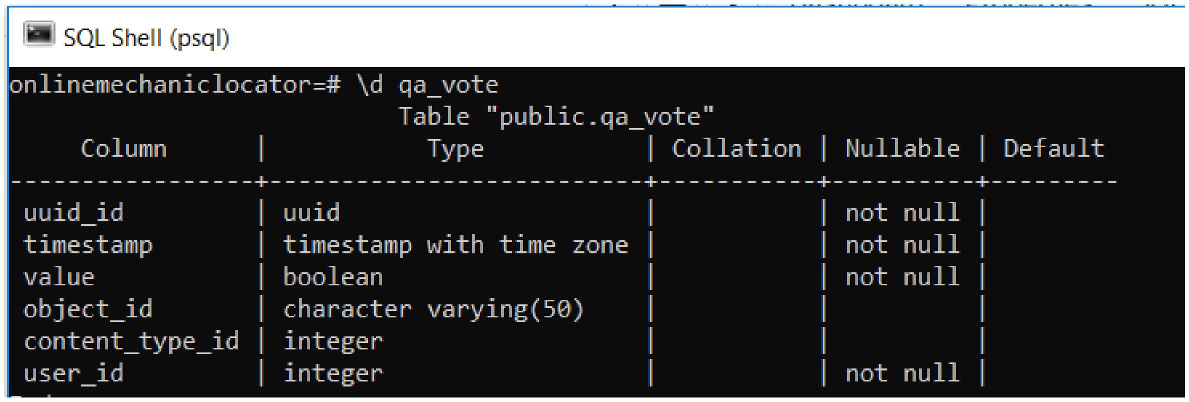
Table 4.5 Question Table



SQL Shell (psql)

```
onlinemechaniclocator=# \d qa_answer
                          Table "public.qa_answer"
   Column    |           Type           | Collation | Nullable | Default
-------------+--------------------------+-----------+----------+---------
 content     | text                     |           | not null |
 uuid_id     | uuid                     |           | not null |
 total_votes | integer                  |           | not null |
 timestamp   | timestamp with time zone |           | not null |
 is_answer   | boolean                  |           | not null |
 question_id | integer                  |           | not null |
 user_id     | integer                  |           | not null |
```

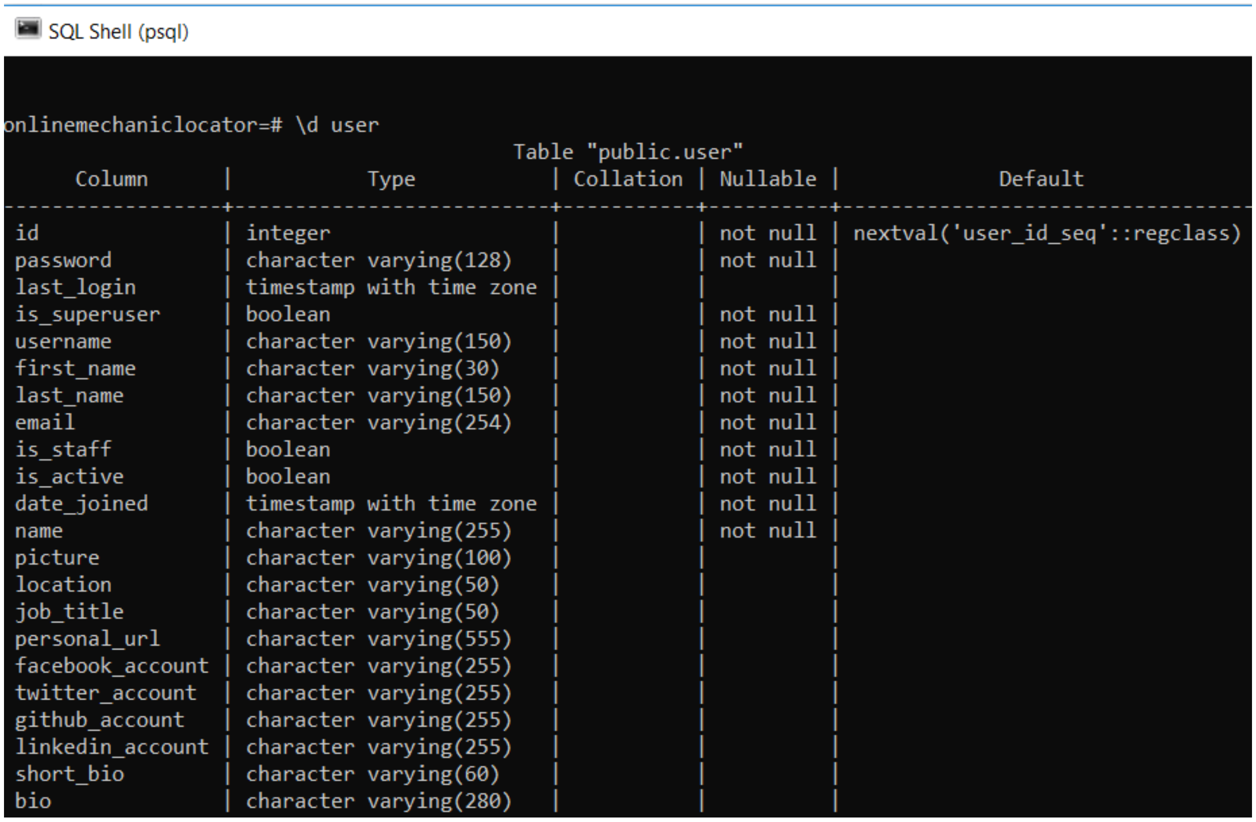
Table 4.6 QA_answer Table



SQL Shell (psql)

```
onlinemechaniclocator=# \d qa_vote
                        Table "public.qa_vote"
     Column      |           Type           | Collation | Nullable | Default
-----------------+--------------------------+-----------+----------+---------
 uuid_id         | uuid                     |           | not null |
 timestamp       | timestamp with time zone |           | not null |
 value           | boolean                  |           | not null |
 object_id       | character varying(50)    |           |          |
 content_type_id | integer                  |           |          |
 user_id         | integer                  |           | not null |
```

Table 4.7 QA_vote Table



SQL Shell (psql)

```
onlinemechaniclocator=# \d user
                                    Table "public.user"
     Column      |           Type           | Collation | Nullable |              Default
-----------------+--------------------------+-----------+----------+----------------------------------
 id              | integer                  |           | not null | nextval('user_id_seq'::regclass)
 password        | character varying(128)   |           | not null |
 last_login      | timestamp with time zone |           |          |
 is_superuser    | boolean                  |           | not null |
 username        | character varying(150)   |           | not null |
 first_name      | character varying(30)    |           | not null |
 last_name       | character varying(150)   |           | not null |
 email           | character varying(254)   |           | not null |
 is_staff        | boolean                  |           | not null |
 is_active       | boolean                  |           | not null |
 date_joined     | timestamp with time zone |           | not null |
 name            | character varying(255)   |           | not null |
 picture         | character varying(100)   |           |          |
 location        | character varying(50)    |           |          |
 job_title       | character varying(50)    |           |          |
 personal_url    | character varying(555)   |           |          |
 facebook_account | character varying(255)  |           |          |
 twitter_account | character varying(255)   |           |          |
 github_account  | character varying(255)   |           |          |
 linkedin_account | character varying(255)  |           |          |
 short_bio       | character varying(60)    |           |          |
 bio             | character varying(280)   |           |          |
```

Table 4.8 User Table

4.2.2 Data Integrity and Constraints

Article Table:

Field Name	Allow Nulls	Unique	Constraint	Key
id	No	Yes	auto-increment	Primary
user	-	-	-	Foreign
content	No	No	-	-
edited	No	No	-	-
image	No	No	-	-
status	No	No	-	-
title	No	No	-	-

User Table:

Field Name	Allow Nulls	Unique	Constraint	Key
id	No	Yes	auto-increment	Primary
password	No	No	-	-
last_login	-	No	-	-
is_superuser	No	No	<=150	-
username	No	Yes	<=30	-
first_name	No	No	<=150	-
last_name	No	No	<=254	-
email	No	Yes	-	-
is_staff	No	No	-	-
is_active	No	No	system-generated	-
date_joined	No	No	-	-
name	No	No	<=255	-
picture	Yes	No	-	-

location	Yes	No	<=50	-
job_title	Yes	No	<=255	-
personal_url	Yes	No	<=255	-
facebook_account	Yes	No	<=255	-
twitter_account	Yes	No	<=255	-
short_bio	Yes	No	<=60	-
bio	Yes	No	<=280	-

Messager Table:

Field Name	Allow Nulls	Unique	Constraint	Key
uuid_id	No	Yes	auto-generated	Foreign
recipient_id	No	-	-	Foreign
sender_id	No	-	-	Foreign
message	No	No	-	-
timestamp	No	No	system_generated	-
unread	No	No	-	-

News Table:

Field Name	Allow Nulls	Unique	Constraint	Key
uuid_id	No	Yes	auto-generated	Primary
parent_id	-	-	-	Foreign
user_id	-	-	-	Foreign
content	No	No	-	-
reply	No	No	system-generated	-
timestamp	No	No	system-generated	-

Qa_Question Table:

Field Name	Allow Nulls	Unique	Constraint	Key
id	No	Yes	auto-increment	Primary
title	No	No	<=200	-
timestamp	No	No	system-generated	-
content	No	No	-	-
status	No	No	<=1	-
has_answer	No	No	-	-
total_votes	No	No	-	-
user_id	No	-	-	Foreign

Offer Table:

Field Name	Allow Nulls	Unique	Constraint	Key
offer_id	No	Yes	auto-increment	Primary
offer_title	No	No	-	-
offer_details	No	No	-	-

Notification Table:

Field Name	Allow Nulls	Unique	Constraint	Key
uuid_id	No	Yes	auto-generated	Primary
actor_id	No	-	-	Foreign
recipient_id	No	-	-	Foreign
timestamp	No	No	system-generated	-
unread	No	No	-	-
verb	No	No	<=1	-

QA_Answer Table

Field Name	Allow Nulls	Unique	Constraint	Key
content	No	No	-	Primary
uuid_id	No	Yes	auto-generated	-
total_votes	No	No	-	-
timestamp	No	No	system-generated	-
is_answer	No	-	-	-
question_id	No	No	-	Foreign
user_id	No	No	-	Foreign

4.3 Procedural Design

4.3.1 Logical Diagram

Use Case:

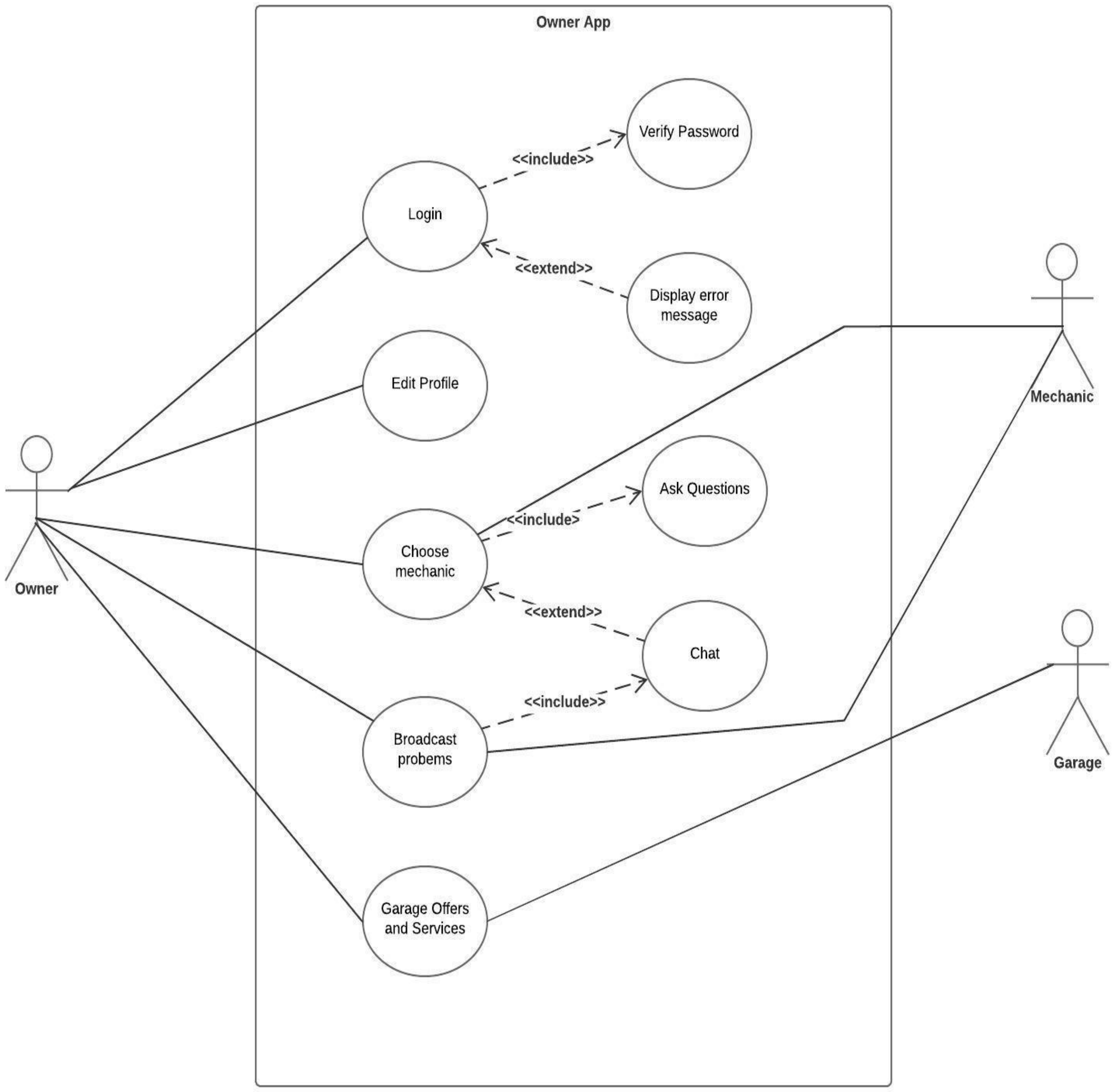


Figure 4.1 Owner App Use Case

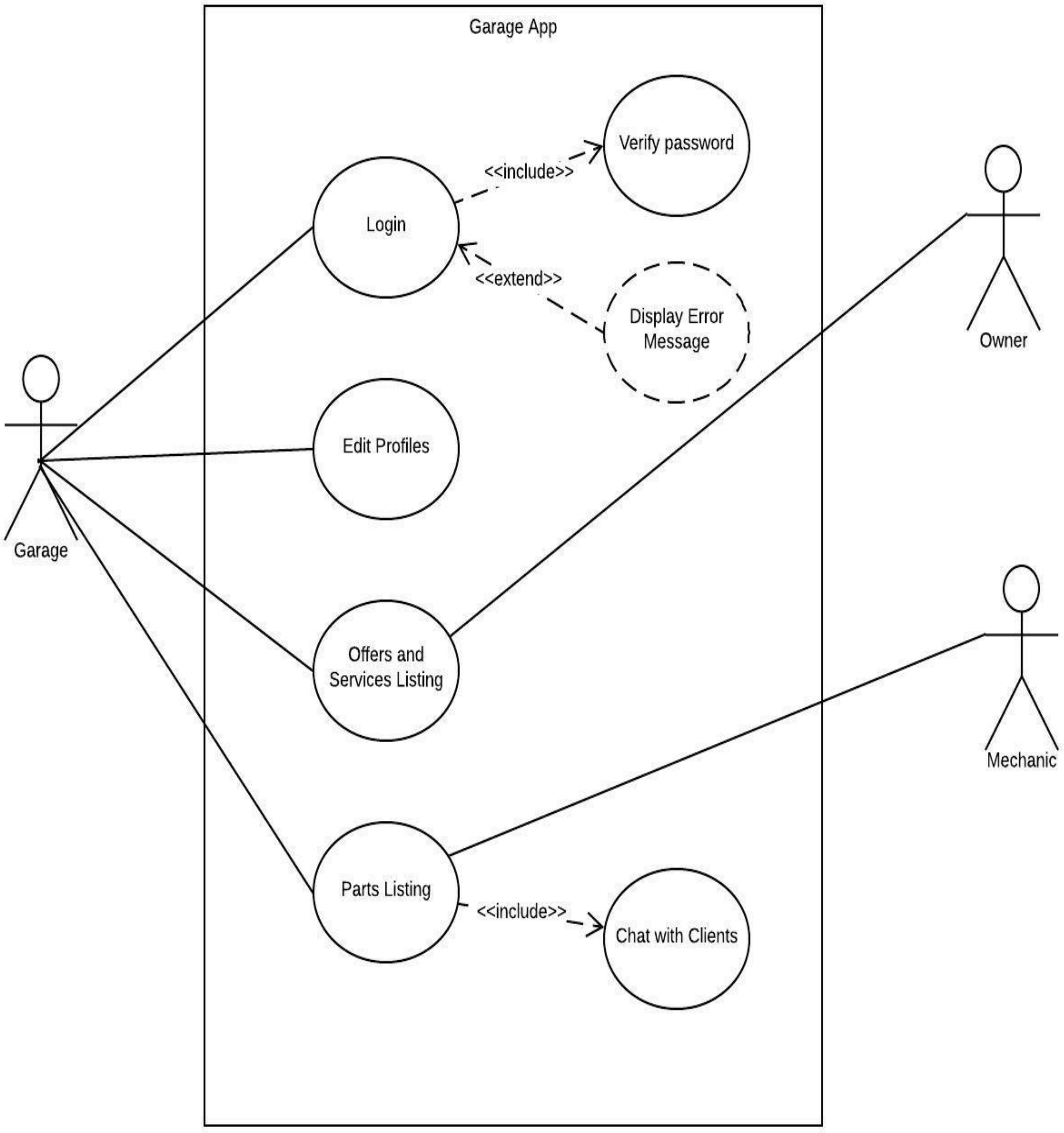


Figure 4.2 Garage App Use Case

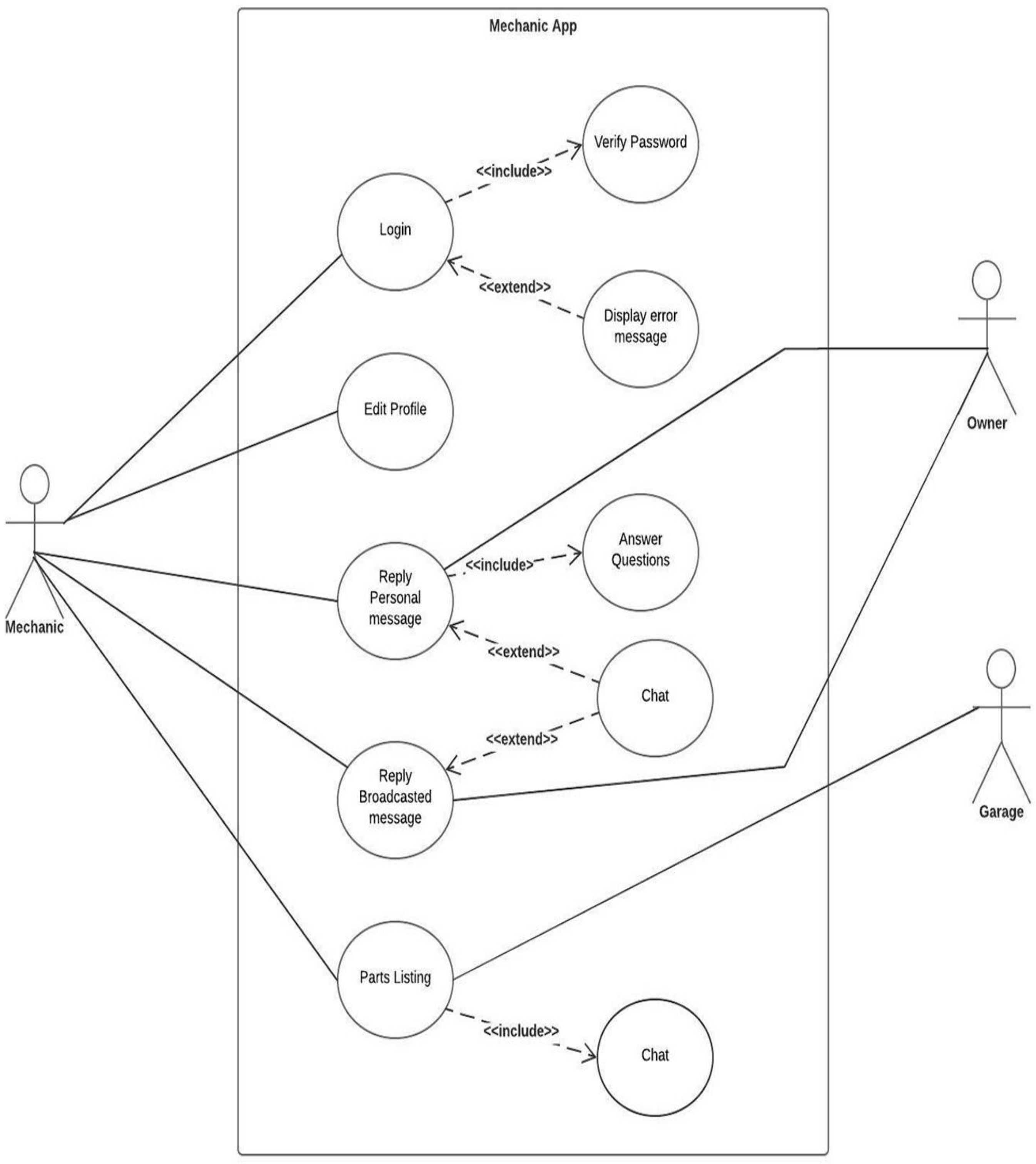


Figure 4.3 Mechanic App Use Case

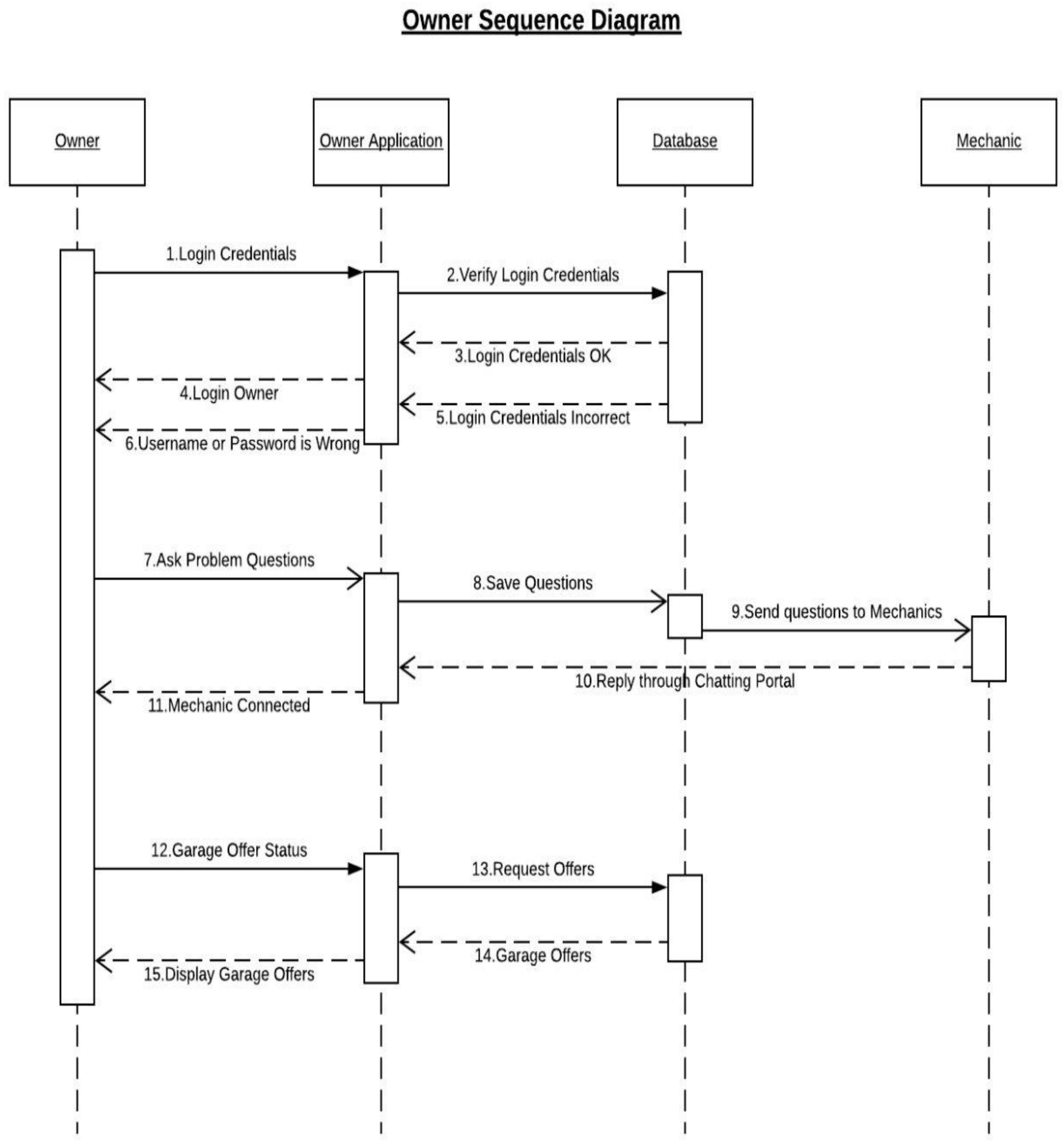


Figure 4.4 Owner Sequence Diagram

Garage Sequence Diagram

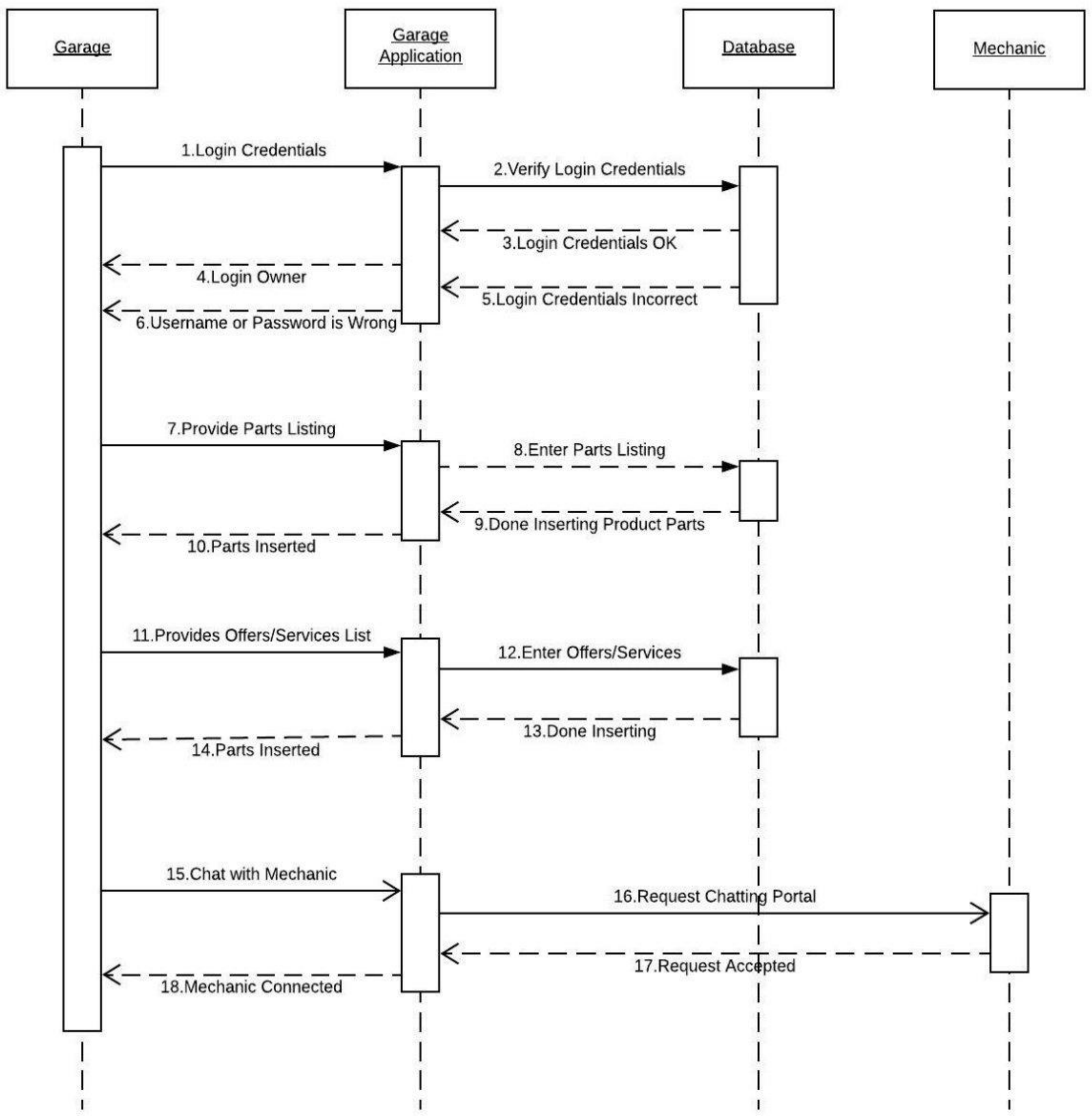


Figure 4.5 Garage Sequence Diagram

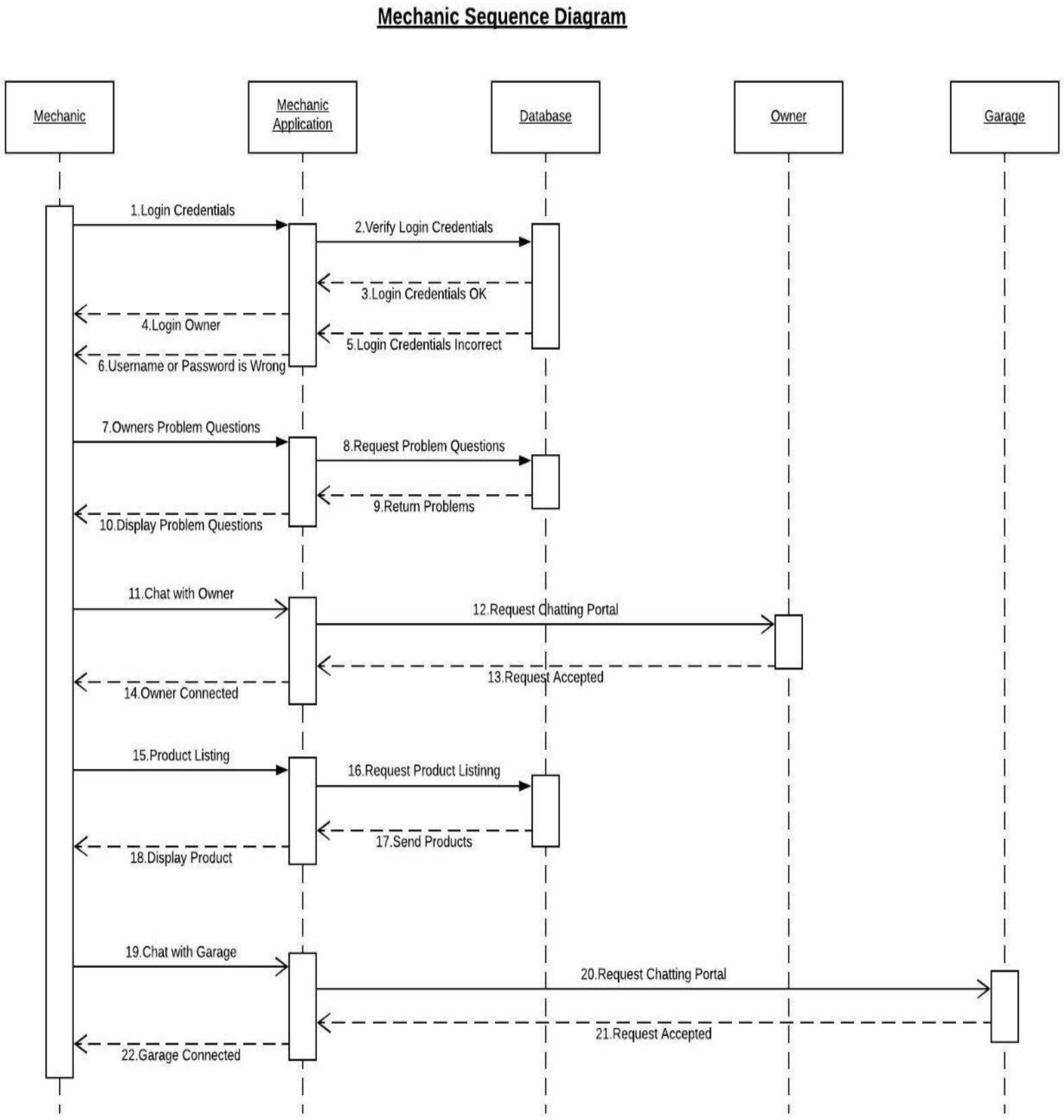


Figure 4.6 Mechanic Sequence Diagram

Activity Diagram:

Garage Activity Diagram:

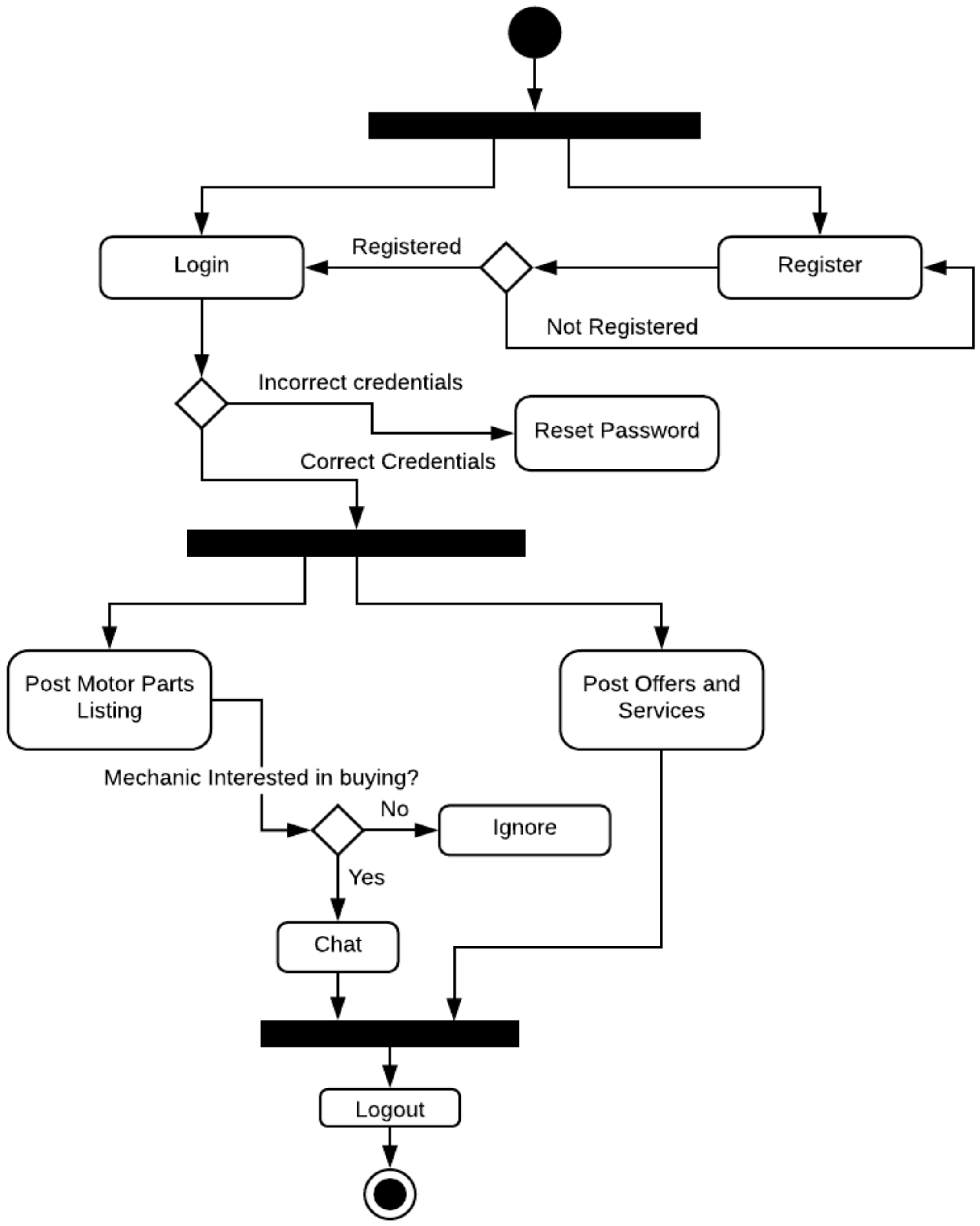


Figure 4.7 Garage Activity Diagram

Mechanic Activity Diagram:

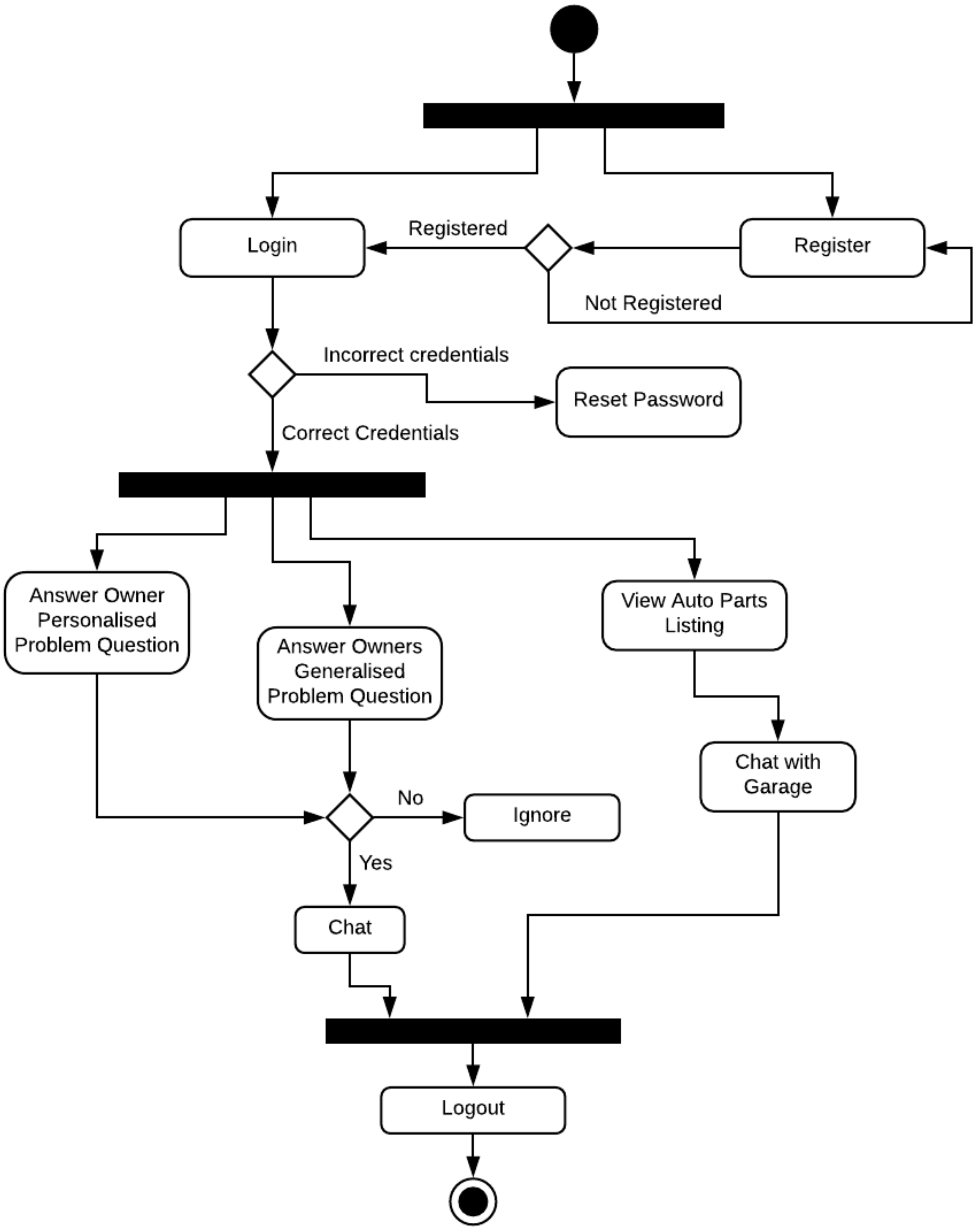


Figure 4.8 Mechanic Activity Diagram

Owner Activity Diagram:

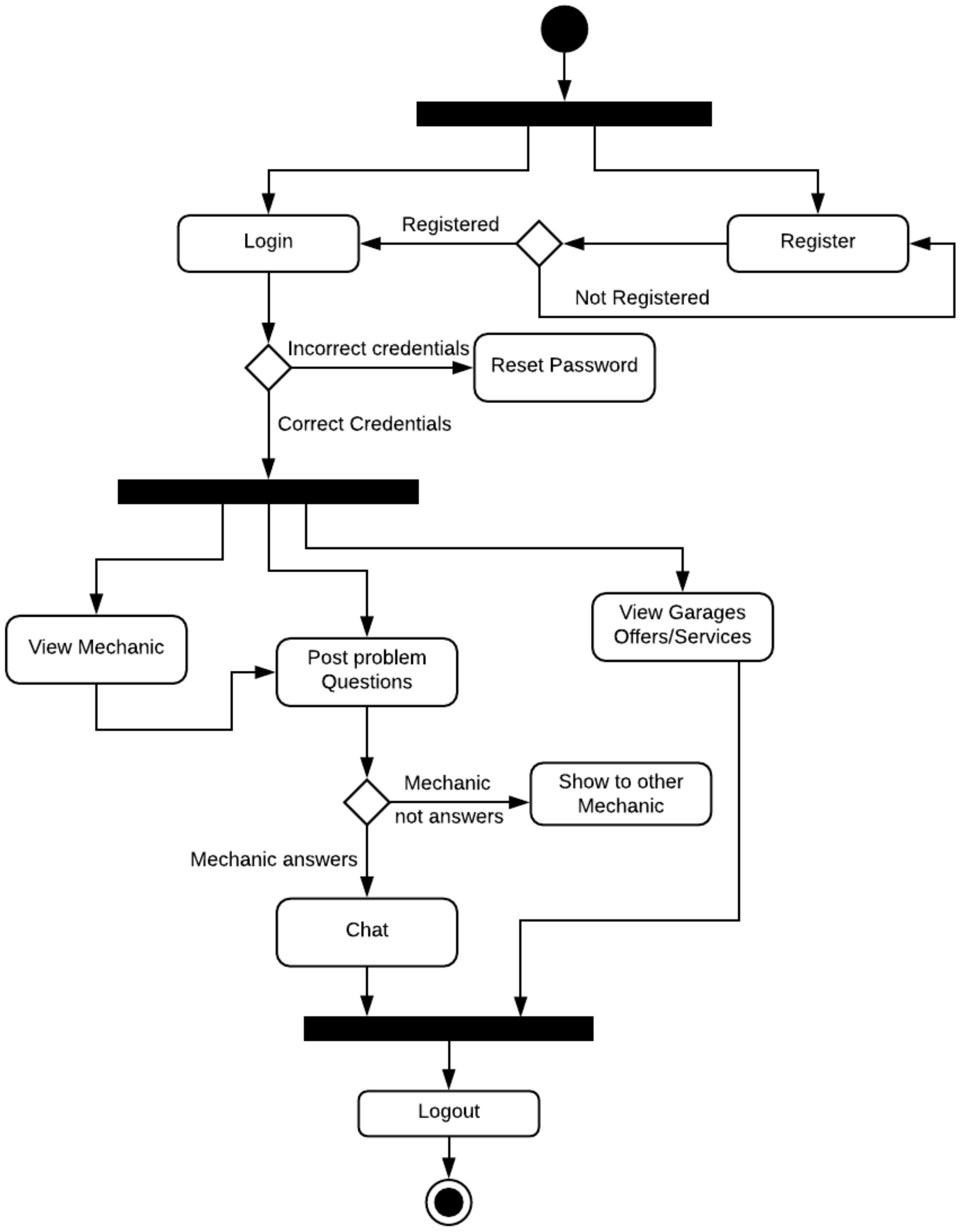
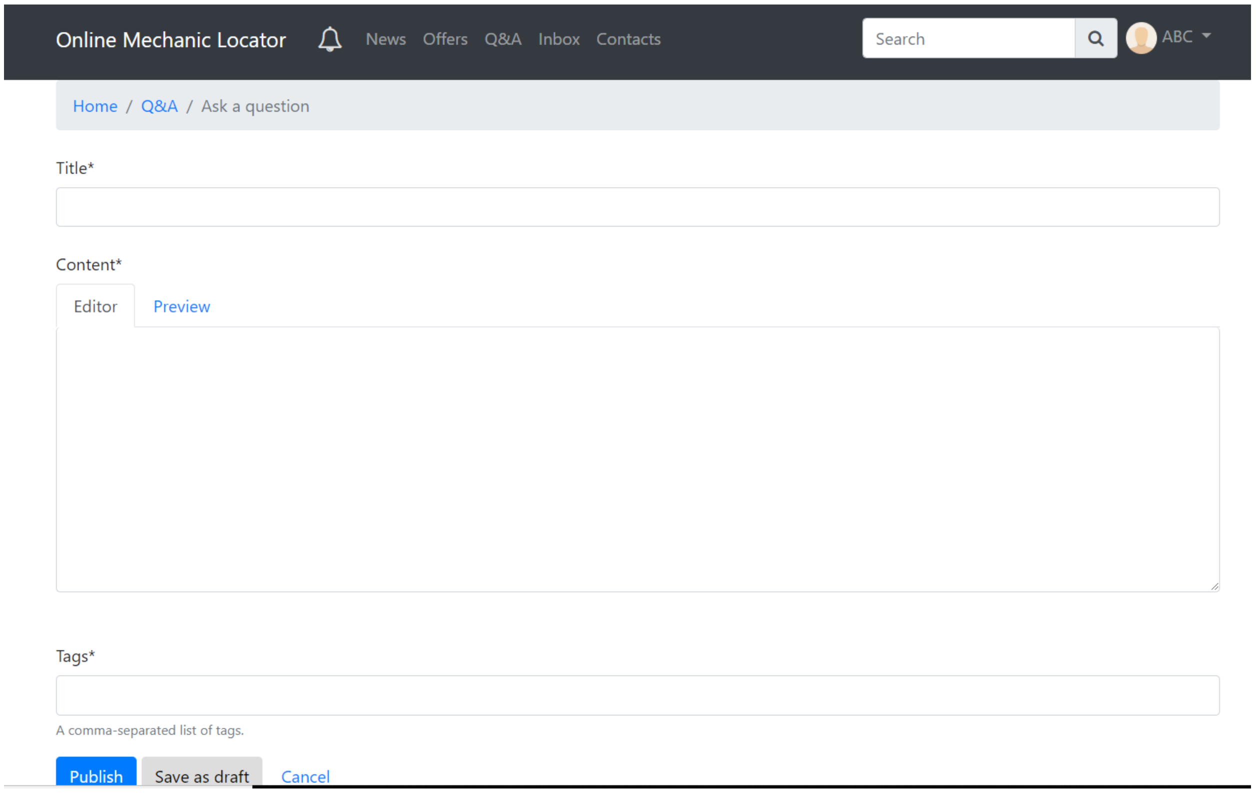
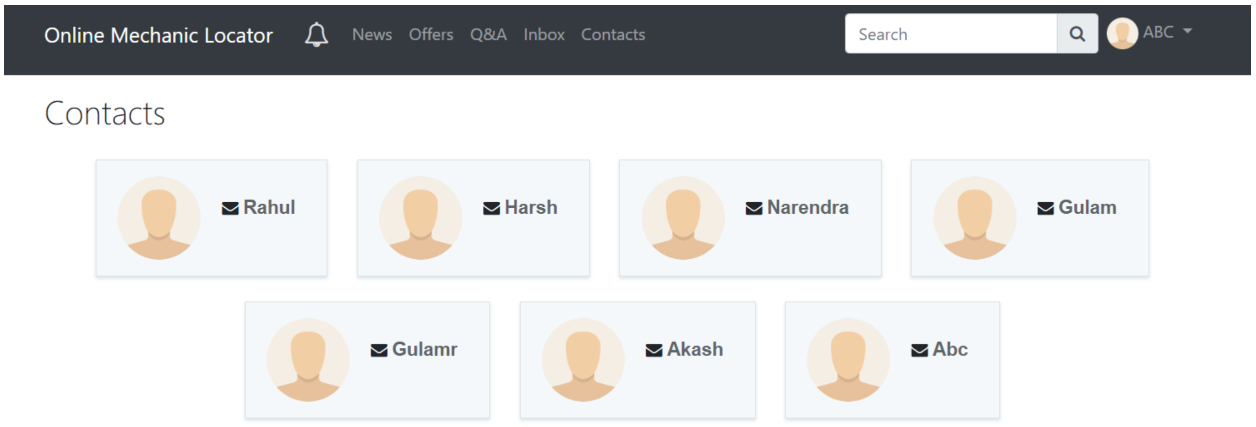


Figure 4.9 Owner Activity Diagram

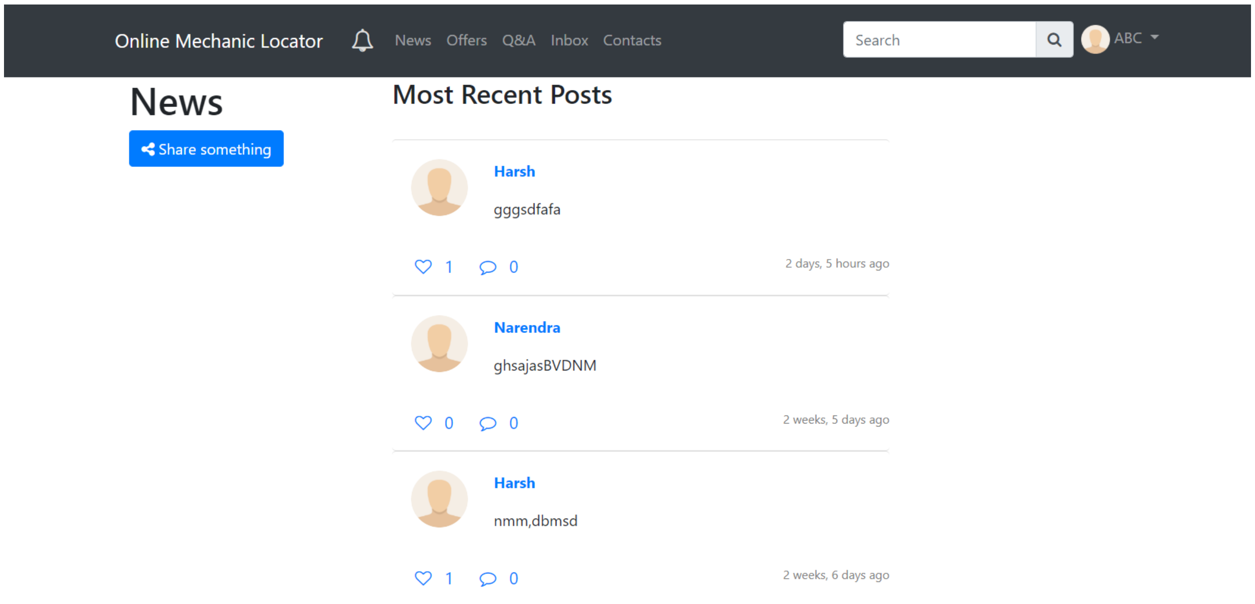
4.4 User Interface Diagram



Broadcast Question Page



Contact Page



Online Mechanic Locator News Offers Q&A Inbox Contacts Search ABC

News

Share something

Most Recent Posts

Harsh

gggsdfafa

1 0 2 days, 5 hours ago

Narendra

ghsajasBVDNM

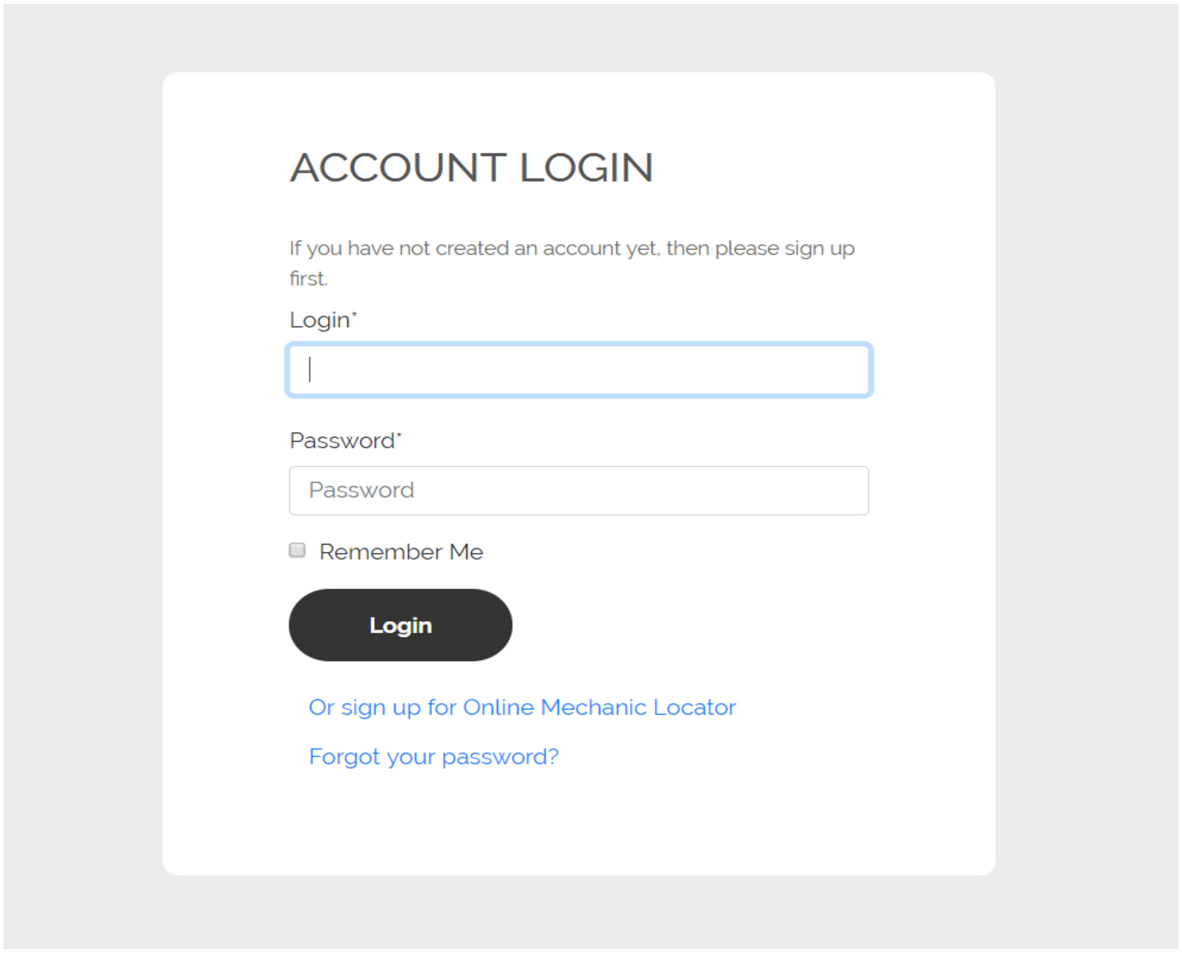
0 0 2 weeks, 5 days ago

Harsh

nmm,dbmsd

1 0 2 weeks, 6 days ago

Home Page



ACCOUNT LOGIN

If you have not created an account yet, then please sign up first.

Login*

Password*

Password

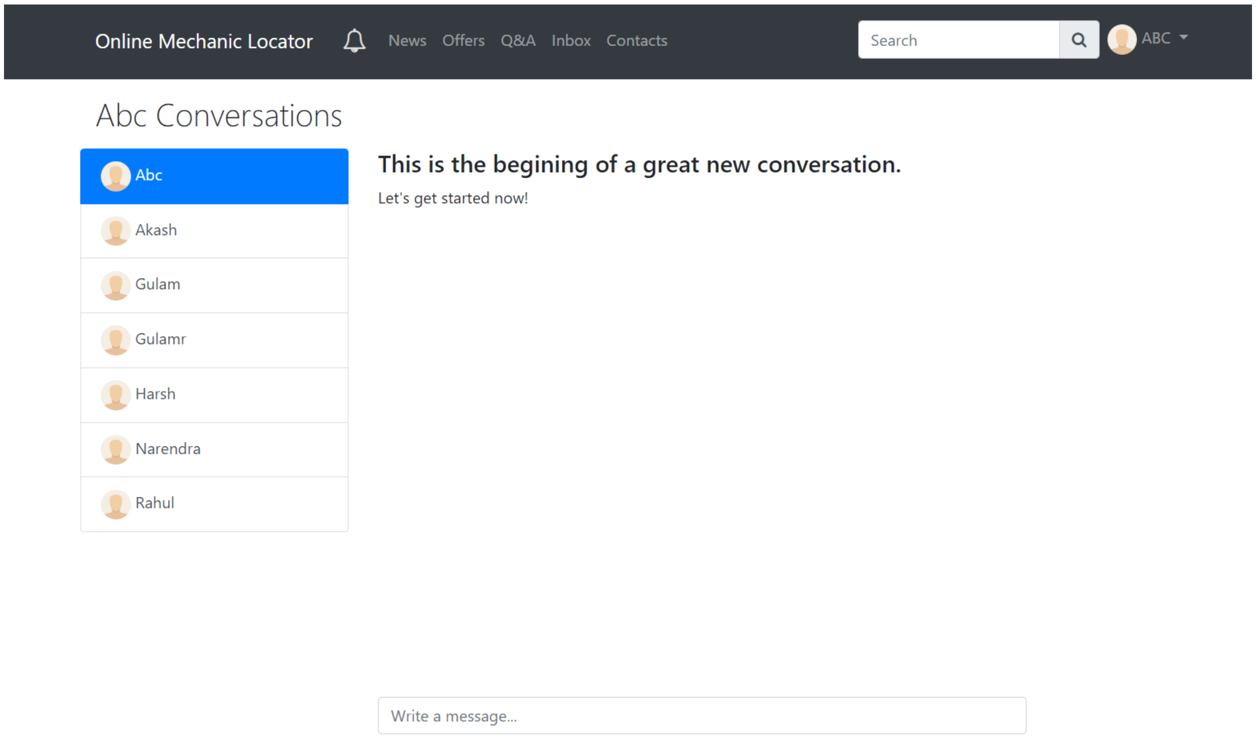
Remember Me

Login

Or sign up for Online Mechanic Locator

Forgot your password?

Login Page



Abc Conversations

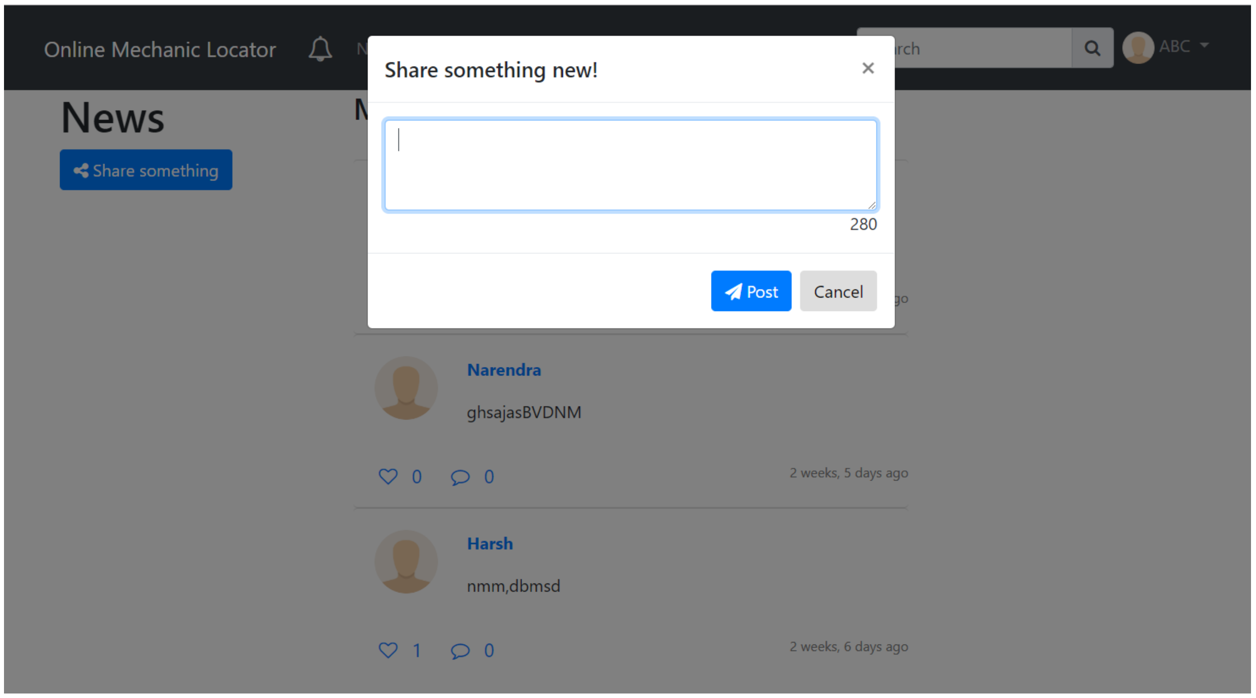
Abc
Akash
Gulam
Gulamr
Harsh
Narendra
Rahul

This is the begining of a great new conversation.

Let's get started now!

Write a message...

Message Page



Narendra

ghsajasBVDNM

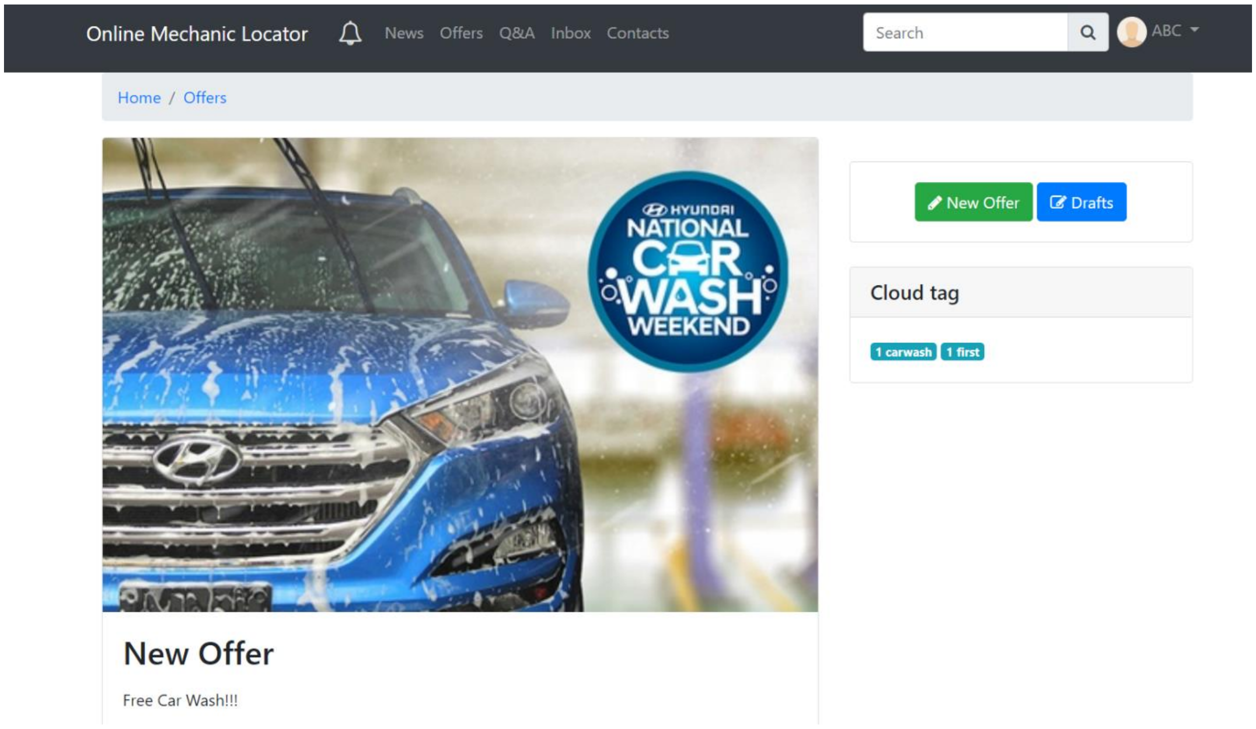
0 0 2 weeks, 5 days ago

Harsh

nmm,dbmsd

1 0 2 weeks, 6 days ago

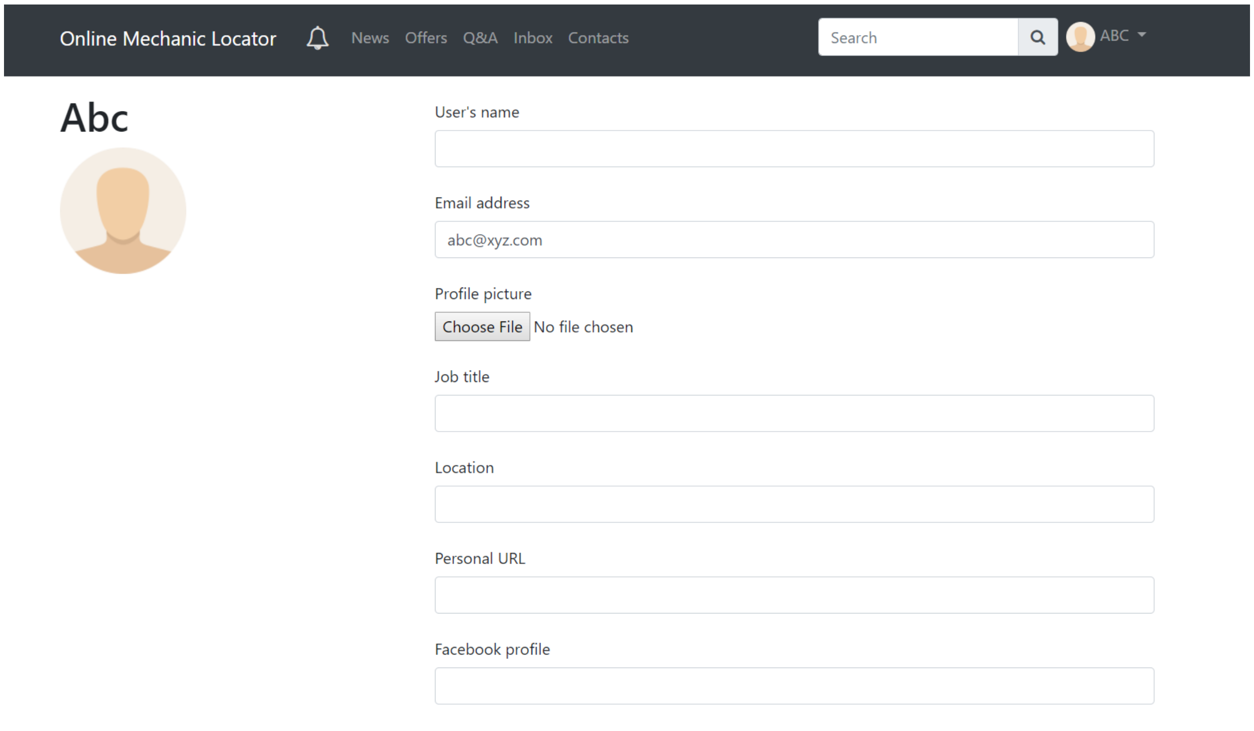
New Sharing Page



New Offer

Free Car Wash!!!

Offer Page



Abc

User's name

Email address: abc@xyz.com

Profile picture: No file chosen

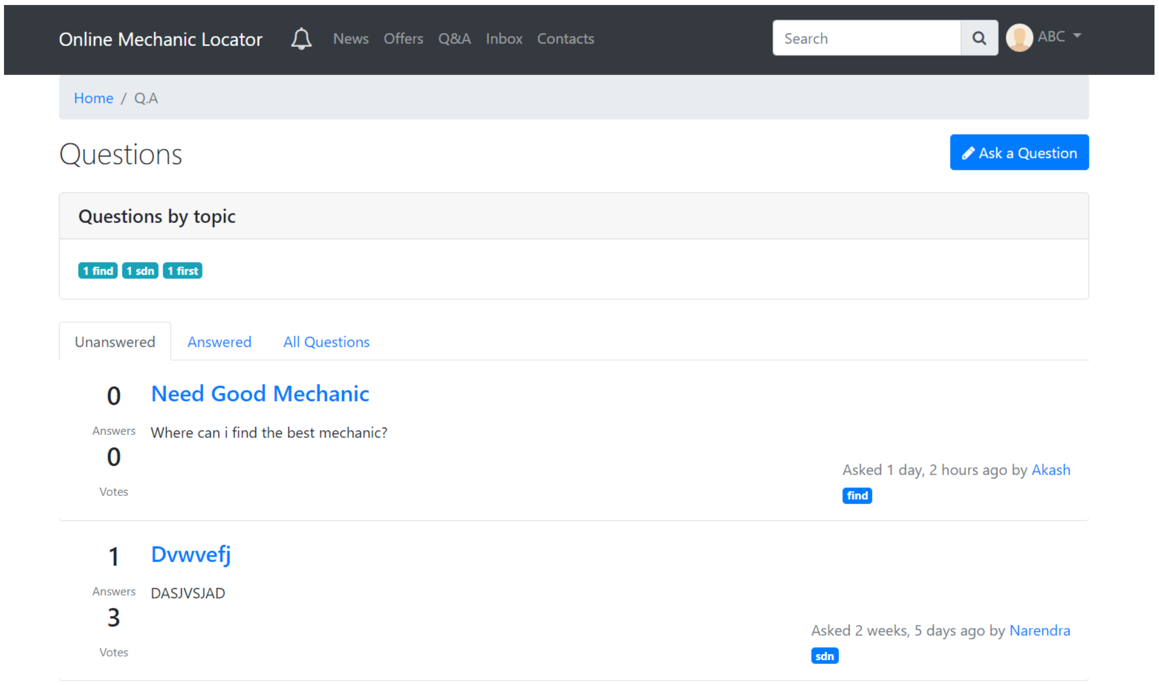
Job title

Location

Personal URL

Facebook profile

Profile Page


Online Mechanic Locator News Offers Q&A Inbox Contacts Search ABC

Home / Q.A

Questions

Ask a Question

Questions by topic

1 find 1 sdn 1 first

Unanswered Answered All Questions

0 Answers 0 Votes

Need Good Mechanic

Where can i find the best mechanic?

Asked 1 day, 2 hours ago by Akash

find

1 Answers 3 Votes

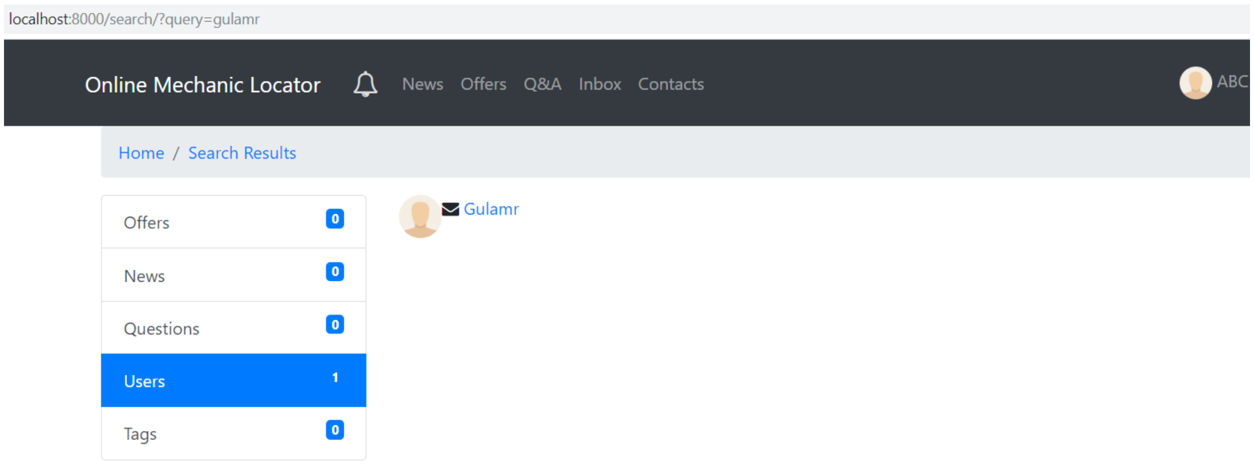
Dvwvefj

DASJVSJAD

Asked 2 weeks, 5 days ago by Narendra

sdn

QnA Page


localhost:8000/search/?query=gulamr

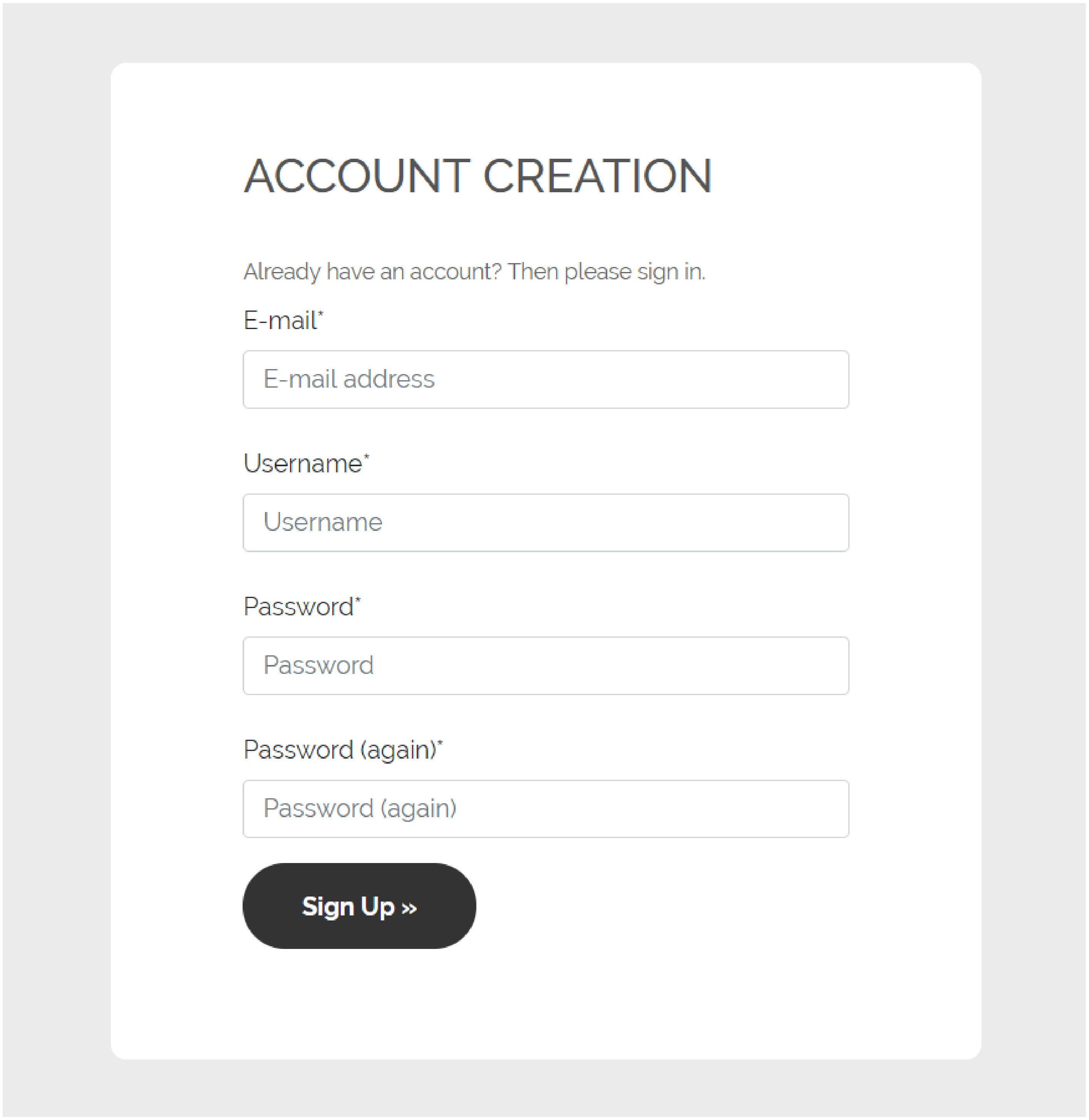
Online Mechanic Locator News Offers Q&A Inbox Contacts ABC

Home / Search Results

Offers	0
News	0
Questions	0
Users	1
Tags	0

Gulamr

Search Page



ACCOUNT CREATION

Already have an account? Then please sign in.

E-mail*

E-mail address

Username*

Username

Password*

Password

Password (again)*

Password (again)

Sign Up Page

Figure 4.10 GUI

4.5 Security

Authentication

- Wrong Username: If the Username is wrong user won't be able to enter the application. The username is checked from the Authentication.
- Wrong Password: If the Password is wrong user won't be able to enter the application. The username is checked from the Authentication.

Profile

- Car Owner Data: The Car owner Data is very important. When the car owner enters the profile, the information is stored in the database. When the car owner enters the app and selects profile, car owner must retain his own data. If the car owner wants to update his profile the data should be updated into database.
- Mechanic Data: When the mechanic enters his profile, the information is stored in the database. When mechanic enters the app and selects profile, mechanic must retain his own data. If the mechanic wants to update his profile the data should be updated into database.
- Garage Owner Data: When the Garage Owner enters his profile, the information is stored in the database. When the garage owner enters the app and selects profile, garage owner must retain his own data. If the garage owner wants to update his profile the data should be update into the database.

Broadcasting Question Problem

- Unauthorized Access: Only the car owners is allowed to broadcast any question problem and the same would be uploaded into the database. Only the car owner is allowed to edit or delete his/her broadcasted question problem. No other person have the access to delete them or edit them**.**
  - The Mechanic has the access only to answer those broadcasted question.
  - The Mechanic cannot chat the car owner until both the party have a deal.

Garage Offers

- Unauthorized Access: Only the garage owner is allowed to upload any offers of their garage into the database.
  - The Mechanic has the access only to check the offers
  - The Mechanic cannot edit the offers only the Garage owners has access to it.

4.6 Test Cases

ID		Name	Description	Steps	Expected outcome
TC1		Login test case	To test the Functionality Of the user	1. Login with the Registered user in the Database 2. Login with partially Correct information 3. Login with no information.	The application Should successfully Login the user when Both email and password Are correct otherwise Display appropriate Error message to the user
TC2		Reset Password Test case	To test Weather user can reset password if the user forgets it	1. Enter email address and the application should click on reset password, be able to send reset 2. Open user email and password email to click on specified link. The user and update 3. Input the new password. The password upon 4. Login with the new password rest. Password.	The application should be able to send reset password email to the user and update the password up on password reset.
TC3		Change Password Test case	To test whether user can change password if user wishes to change	1. Login using the user credentials. 2. Open user profile and open change password dialog. 3. Input current password and new password twice. 4. Logout and try to login with new password	The application will be able to change user's password if the user knows it's current password, the same should reflect when user tries to login next time.
TC4		Broadcasting Question Problem Test case	To test whether the user can broadcast a problem question	1. Login using the user credentials. 2. In home page, select Broadcast option and type the problem question in the text field.	The application will be able to broadcast problem questions which user inputs.

				3. Click on Send button to broadcast it.	
TC5		Logout test case	To test the functionality of logging out of the user	1. Click on the Gear icon 2. Scroll & Click on the logout button.	The application should be able to logout the current user, and display login screen.

Table 4.9 Test Cases

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 Implementation Approaches

We followed Iterative Model Approach. Since, the Iterative model is repetition incarnate. Instead of starting with fully known requirements, you implement a set of software requirements, then test, evaluate and pinpoint further requirements. A new version of the software is produced with each phase, or iteration. Rinse and repeat until the complete system are ready.
Python web Framework Django is used to make this project. Since Django uses Model View Template (MVT) approach, the project follows MVT approach for application building.
Model – Database Schema is defined in form of python classes
View – Here your Application Logic is written
Template – These files contain all the code for look and feel of the website

Django also provides an API to work with database. The supported Object Relational Mapper (ORM) can be used so that the database schema defined as python classes in models.py can be converted into appropriate SQL queries which can be run on Database.
Web Application Development with Django becomes easy because Django follows a very simplified approach for solving the web development problems which is to divide the bigger problem in smaller ones so that it can be solved easily and also the DRY(DO NOT REPEAT YOURSELF) principles is followed by using this approach.

5.1.1 Implementation Plan

The website is made of various other components that works together to make it function properly. The following components have been developed to fulfill the requirements of this project.

1. **Offers** - This component is responsible for offer content on website
2. **Messager** – This component can be used so that users on the application can chat with each other
3. **News** – This component is used so that uses on the application can share exciting news.
4. Notifications – This I component is responsible for sending real-time push notification to the users.
5. **QA** – This component is used to provide qna functionality so that users can interact with each other.
6. **Search** – This component is responsible for providing search functionality.
7. **User** – This component is used to store user Information

The plan is to develop these component iteratively and then to combine them to make complete Online Mechanic Locator Web Application.

5.1.2 Implementation Standards

PEP 8, sometimes spelled PEP8 or PEP-8, is a document that provides guidelines and best practices on how to write Python code. It was written in 2001 by Guido van Rossum, Barry Warsaw, and Nick Coghlan. The primary focus of PEP 8 is to improve the readability and consistency of Python code.

PEP stands for Python Enhancement Proposal, and there are several of them. A PEP is a document that describes new features proposed for Python and documents aspects of Python, like design and style, for the community.
PEP 8 standard is followed throughout this project.

5.2 Coding Details and Efficiency

Codes for some important components are shown here. CSS and HTML code are quite big, so they won't be included in the blackbook. For full codebase, please refer to the GitHub repository for this project i.e., https://github.com/akashverma975/Online-Mechanic-Locator or CD which is submitted in the college.

QnA:

```
import uuid
from collections import Counter

from django.conf import settings
from     django.contrib.contenttypes.fields     import     GenericForeignKey,
GenericRelation
from django.contrib.contenttypes.models import ContentType
from django.db import models
from django.db.models import Count
from django.utils.translation import ugettext_lazy as _

from slugify import slugify

from taggit.managers import TaggableManager
from markdownx.models import MarkdownxField
from markdownx.utils import markdownify


class Vote(models.Model):
    """Model class to host every vote, made with ContentType framework to
    allow a single model connected to Questions and Answers."""
    uuid_id = models.UUIDField(
        primary_key=True, default=uuid.uuid4, editable=False)
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL, on_delete=models.CASCADE)
    timestamp = models.DateTimeField(auto_now_add=True)
    value = models.BooleanField(default=True)
    content_type = models.ForeignKey(ContentType,
        blank=True,            null=True,            related_name="votes_on",
on_delete=models.CASCADE)
    object_id = models.CharField(
        max_length=50, blank=True, null=True)
    vote = GenericForeignKey(
        "content_type", "object_id")

    class Meta:
        verbose_name = _("Vote")
        verbose_name_plural = _("Votes")
        index_together = ("content_type", "object_id")
        unique_together = ("user", "content_type", "object_id")


class QuestionQuerySet(models.query.QuerySet):
    """Personalized queryset created to improve model usability"""

    def get_answered(self):
        """Returns only items which has been marked as answered in the current
        queryset"""
        return self.filter(has_answer=True)

    def get_unanswered(self):
        """Returns only items which has not been marked as answered in the
        current queryset"""
        return self.filter(has_answer=False)

    def get_counted_tags(self):
        """Returns a dict element with tags and its count to show on the UI."""
        tag_dict = {}
        query = self.all().annotate(tagged=Count('tags')).filter(tags__gt=0)
        for obj in query:
            for tag in obj.tags.names():
                if tag not in tag_dict:
                    tag_dict[tag] = 1

                else:  # pragma: no cover
                    tag_dict[tag] += 1

        return tag_dict.items()


class Question(models.Model):
    """Model class to contain every question in the forum."""
    OPEN = "O"
    CLOSED = "C"
    DRAFT = "D"
    STATUS = (
        (OPEN, _("Open")),
        (CLOSED, _("Closed")),
        (DRAFT, _("Draft")),
    )
    user            =            models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    title = models.CharField(max_length=200, unique=True, blank=False)
    timestamp = models.DateTimeField(auto_now_add=True)
    slug = models.SlugField(max_length=80, null=True, blank=True)
    status = models.CharField(max_length=1, choices=STATUS, default=DRAFT)
    content = MarkdownxField()
    has_answer = models.BooleanField(default=False)
    total_votes = models.IntegerField(default=0)
    votes = GenericRelation(Vote)
    tags = TaggableManager()
    objects = QuestionQuerySet.as_manager()

    class Meta:
        ordering = ["-timestamp"]
        verbose_name = _("Question")
        verbose_name_plural = _("Questions")

    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(f"{self.title}-{self.id}",
                                to_lower=True, max_length=80)

        super().save(*args, **kwargs)

    def __str__(self):
        return self.title

    @property
    def count_answers(self):
        return Answer.objects.filter(question=self).count()

    def count_votes(self):
        """Method to update the sum of the total votes. Uses this complex
query
        to avoid race conditions at database level."""
        dic = Counter(self.votes.values_list("value", flat=True))
        Question.objects.filter(id=self.id).update(total_votes=dic[True]    -
dic[False])
        self.refresh_from_db()

    def get_upvoters(self):
        """Returns a list containing the users who upvoted the instance."""
        return [vote.user for vote in self.votes.filter(value=True)]

    def get_downvoters(self):
        """Returns a list containing the users who downvoted the instance."""
        return [vote.user for vote in self.votes.filter(value=False)]
    def get_answers(self):
        return Answer.objects.filter(question=self)

    def get_accepted_answer(self):
        return Answer.objects.get(question=self, is_answer=True)

    def get_markdown(self):
        return markdownify(self.content)


class Answer(models.Model):
    """Model class to contain every answer in the forum and to link it
    to its respective question."""
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    user            =            models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE)
    content = MarkdownxField()
    uuid_id = models.UUIDField(
        primary_key=True, default=uuid.uuid4, editable=False)
    total_votes = models.IntegerField(default=0)
    timestamp = models.DateTimeField(auto_now_add=True)
    is_answer = models.BooleanField(default=False)
    votes = GenericRelation(Vote)

    class Meta:
        ordering = ["-is_answer", "-timestamp"]
        verbose_name = _("Answer")
        verbose_name_plural = _("Answers")

    def __str__(self):  # pragma: no cover
        return self.content

    def get_markdown(self):
        return markdownify(self.content)

    def count_votes(self):
        """Method to update the sum of the total votes. Uses this complex
query
        to avoid race conditions at database level."""
        dic = Counter(self.votes.values_list("value", flat=True))

Answer.objects.filter(uuid_id=self.uuid_id).update(total_votes=dic[True]    -
dic[False])
        self.refresh_from_db()

    def get_upvoters(self):
        """Returns a list containing the users who upvoted the instance."""
        return [vote.user for vote in self.votes.filter(value=True)]

    def get_downvoters(self):
        """Returns a list containing the users who downvoted the instance."""
        return [vote.user for vote in self.votes.filter(value=False)]

    def accept_answer(self):
        answer_set = Answer.objects.filter(question=self.question)
        answer_set.update(is_answer=False)
        self.is_answer = True
        self.save()
        self.question.has_answer = True
        self.question.save()
```

Table 5.1 QnA models.py

```
from django.db.utils import IntegrityError
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.contrib import messages
from django.http import JsonResponse
from django.urls import reverse
from django.utils.translation import ugettext as _
from django.views.decorators.http import require_http_methods
from django.views.generic import CreateView, ListView, DetailView

from onlinemechaniclocator.helpers import ajax_required
from onlinemechaniclocator.qa.models import Question, Answer
from onlinemechaniclocator.qa.forms import QuestionForm


class QuestionsIndexListView(LoginRequiredMixin, ListView):
    """CBV to render a list view with all the registered questions."""
    model = Question
    paginate_by = 20
    context_object_name = "questions"

    def get_context_data(self, *args, **kwargs):
        context = super().get_context_data(*args, **kwargs)
        context["popular_tags"] = Question.objects.get_counted_tags()
        context["active"] = "all"
        return context


class QuestionAnsListView(QuestionsIndexListView):
    """CBV to render a list view with all question which have been already
    marked as answered."""

    def get_queryset(self, **kwargs):
        return Question.objects.get_answered()

    def get_context_data(self, *args, **kwargs):
        context = super().get_context_data(*args, **kwargs)
        context["active"] = "answered"
        return context


class QuestionListView(QuestionsIndexListView):
    """CBV to render a list view with all question which haven't been marked
    as answered."""

    def get_queryset(self, **kwargs):
        return Question.objects.get_unanswered()

    def get_context_data(self, *args, **kwargs):
        context = super().get_context_data(*args, **kwargs)
        context["active"] = "unanswered"
        return context


class QuestionDetailView(LoginRequiredMixin, DetailView):
    """View to call a given Question object and to render all the details
about
    that Question."""
    model = Question
    context_object_name = "question"


class CreateQuestionView(LoginRequiredMixin, CreateView):
    """
    View to handle the creation of a new question
    """
    form_class = QuestionForm
    template_name = "qa/question_form.html"
    message = _("Your question has been created.")

    def form_valid(self, form):
        form.instance.user = self.request.user
        return super().form_valid(form)

    def get_success_url(self):
        messages.success(self.request, self.message)
        return reverse("qa:index_noans")


class CreateAnswerView(LoginRequiredMixin, CreateView):
    """
    View to create new answers for a given question
    """
    model = Answer
    fields = ["content", ]
    message = _("Thank you! Your answer has been posted.")

    def form_valid(self, form):
        form.instance.user = self.request.user
        form.instance.question_id = self.kwargs["question_id"]
        return super().form_valid(form)

    def get_success_url(self):
        messages.success(self.request, self.message)
        return reverse(
            "qa:question_detail", kwargs={"pk": self.kwargs["question_id"]})


@login_required
@ajax_required
@require_http_methods(["POST"])
def question_vote(request):
    """Function view to receive AJAX call, returns the count of votes a given
    question has recieved."""
    question_id = request.POST["question"]
    value = None
    if request.POST["value"] == "U":
        value = True

    else:
        value = False

    question = Question.objects.get(pk=question_id)
    try:
        question.votes.update_or_create(
            user=request.user, defaults={"value": value}, )
        question.count_votes()
        return JsonResponse({"votes": question.total_votes})

    except IntegrityError:  # pragma: no cover
        return JsonResponse({'status': 'false',
                             'message': _("Database integrity error.")},
                            status=500)


@login_required
@ajax_required
@require_http_methods(["POST"])
def answer_vote(request):
    """Function view to receive AJAX call, returns the count of votes a given
    answer has recieved."""
    answer_id = request.POST["answer"]
    value = None
    if request.POST["value"] == "U":
        value = True

    else:
        value = False

    answer = Answer.objects.get(uuid_id=answer_id)
    try:
        answer.votes.update_or_create(
            user=request.user, defaults={"value": value}, )
        answer.count_votes()
        return JsonResponse({"votes": answer.total_votes})

    except IntegrityError:  # pragma: no cover
        return JsonResponse({'status': 'false',
                              'message': _("Database integrity error.")},
                             status=500)


@login_required
@ajax_required
@require_http_methods(["POST"])
def accept_answer(request):
    """Function view to receive AJAX call, marks as accepted a given answer
for
    an also provided question."""
    answer_id = request.POST["answer"]
    answer = Answer.objects.get(uuid_id=answer_id)
    answer.accept_answer()
    return JsonResponse({'status': 'true'}, status=200)
```

Table 5.2 QnA views.py

```
from django.conf.urls import url

from onlinemechaniclocator.qa import views

app_name = 'qa'
urlpatterns = [
    url(r'^$', views.QuestionListView.as_view(), name='index_noans'),
    url(r'^answered/$',                  views.QuestionAnsListView.as_view(),
name='index_ans'),
    url(r'^indexed/$',                views.QuestionsIndexListView.as_view(),
name='index_all'),
    url(r'^question-detail/(?P<pk>\d+)/$',
views.QuestionDetailView.as_view(), name='question_detail'),
    url(r'^ask-question/$',               views.CreateQuestionView.as_view(),
name='ask_question'),
    url(r'^propose-answer/(?P<question_id>\d+)/$',
views.CreateAnswerView.as_view(), name='propose_answer'),
    url(r'^question/vote/$', views.question_vote, name='question_vote'),
    url(r'^answer/vote/$', views.answer_vote, name='answer_vote'),
    url(r'^accept-answer/$', views.accept_answer, name='accept_answer'),
]
```

Table 5.3 QnA urls.py

Search:

```
from django.contrib.auth import get_user_model
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.db.models import Q
from django.http import JsonResponse
from django.views.generic import ListView

from taggit.models import Tag

from onlinemechaniclocator.articles.models import Article
from onlinemechaniclocator.news.models import News
from onlinemechaniclocator.helpers import ajax_required
from onlinemechaniclocator.qa.models import Question


class SearchListView(LoginRequiredMixin, ListView):
    """CBV to contain all the search results"""
    model = News
    template_name = "search/search_results.html"

    def get_context_data(self, *args, **kwargs):
        context = super().get_context_data(*args, **kwargs)
        query = self.request.GET.get("query")
        context["active"] = 'news'
        context["hide_search"] = True
        context["tags_list"] = Tag.objects.filter(name=query)
        context["news_list"] = News.objects.filter(
            content__icontains=query, reply=False)
        context["articles_list"] = Article.objects.filter(Q(
            title__icontains=query) | Q(content__icontains=query) | Q(
                tags__name__icontains=query), status="P")
        context["questions_list"] = Question.objects.filter(
            Q(title__icontains=query) | Q(content__icontains=query) | Q(
                tags__name__icontains=query))
        context["users_list"] = get_user_model().objects.filter(
            Q(username__icontains=query) | Q(
                name__icontains=query))
        context["news_count"] = context["news_list"].count()
        context["articles_count"] = context["articles_list"].count()
        context["questions_count"] = context["questions_list"].count()
        context["users_count"] = context["users_list"].count()
        context["tags_count"] = context["tags_list"].count()
        context["total_results"] = context["news_count"] + \
            context["articles_count"] + context["questions_count"] + \
            context["users_count"] + context["tags_count"]
        return context


# For autocomplete suggestions
@login_required
@ajax_required
```

```
def get_suggestions(request):
    # Convert users, articles, questions objects into list to be
    # represented as a single list.
    query = request.GET.get('term', '')
    users = list(get_user_model().objects.filter(
        Q(username__icontains=query) | Q(name__icontains=query)))
    articles = list(Article.objects.filter(
        Q(title__icontains=query) | Q(content__icontains=query) | Q(
            tags__name__icontains=query), status="P"))
    questions = list(Question.objects.filter(Q(title__icontains=query) | Q(
        content__icontains=query) | Q(tags__name__icontains=query)))
    # Add all the retrieved users, articles, questions to data_retrieved
    # list.
    data_retrieved = users
    data_retrieved.extend(articles)
    data_retrieved.extend(questions)
    results = []
    for data in data_retrieved:
        data_json = {}
        if isinstance(data, get_user_model()):
            data_json['id'] = data.id
            data_json['label'] = data.username
            data_json['value'] = data.username

        if isinstance(data, Article):
            data_json['id'] = data.id
            data_json['label'] = data.title
            data_json['value'] = data.title

        if isinstance(data, Question):
            data_json['id'] = data.id
            data_json['label'] = data.title
            data_json['value'] = data.title

        results.append(data_json)

    return JsonResponse(results, safe=False)
```

Table 5.4 search views.py

```
from django.conf.urls import url

from onlinemechaniclocator.search import views

app_name = 'search'
urlpatterns = [
    url(r'^$', views.SearchListView.as_view(), name='results'),
    url(r'^suggestions/$', views.get_suggestions, name='suggestions'),
]
```

Table 5.5 search urls.py

Notification:

```
import uuid

from django.conf import settings
from django.contrib.auth import get_user_model
from django.contrib.contenttypes.fields import GenericForeignKey
from django.contrib.contenttypes.models import ContentType
from django.core import serializers
from django.db import models
from django.utils.translation import ugettext_lazy as _

from asgiref.sync import async_to_sync

from channels.layers import get_channel_layer

from slugify import slugify


class NotificationQuerySet(models.query.QuerySet):
    """Personalized queryset created to improve model usability"""

    def unread(self):
        """Return only unread items in the current queryset"""
        return self.filter(unread=True)

    def read(self):
        """Return only read items in the current queryset"""
        return self.filter(unread=False)

    def mark_all_as_read(self, recipient=None):
        """Mark as read any unread elements in the current queryset with
        optional filter by recipient first.
        """
        qs = self.unread()
        if recipient:
            qs = qs.filter(recipient=recipient)

        return qs.update(unread=False)

    def mark_all_as_unread(self, recipient=None):
        """Mark as unread any read elements in the current queryset with
        optional filter by recipient first.
        """
        qs = self.read()
        if recipient:
            qs = qs.filter(recipient=recipient)

        return qs.update(unread=True)

    def serialize_latest_notifications(self, recipient=None):
        """Returns a serialized version of the most recent unread elements
in
        the queryset"""
        qs = self.unread()[:5]
        if recipient:
            qs = qs.filter(recipient=recipient)[:5]

        notification_dic = serializers.serialize("json", qs)
        return notification_dic

    def get_most_recent(self, recipient=None):
        """Returns the most recent unread elements in the queryset"""
        qs = self.unread()[:5]
        if recipient:
            qs = qs.filter(recipient=recipient)[:5]

        return qs


class Notification(models.Model):
    """
    Action model describing the actor acting out a verb (on an optional
target).
    Nomenclature based on http://activitystrea.ms/specs/atom/1.0/

    This model is an adaptation from the django package django-notifications
at
    https://github.com/django-notifications/django-notifications

    Generalized Format::

        <actor> <verb> <time>
        <actor> <verb> <action_object> <time>

    Examples::

        <Sebastian> <Logged In> <1 minute ago>
        <Sebastian> <commented> <Article> <2 hours ago>
    """
    LIKED = 'L'
    COMMENTED = 'C'
    FAVORITED = 'F'
    ANSWERED = 'A'
    ACCEPTED_ANSWER = 'W'
    EDITED_ARTICLE = 'E'
    ALSO_COMMENTED = 'K'
    LOGGED_IN = 'I'
    LOGGED_OUT = 'O'
    VOTED = 'V'
    SHARED = 'S'
    SIGNUP = 'U'
    REPLY = 'R'
    NOTIFICATION_TYPES = (
        (LIKED, _('liked')),
        (COMMENTED, _('commented')),
        (FAVORITED, _('cavorited')),
        (ANSWERED, _('answered')),
        (ACCEPTED_ANSWER, _('accepted')),
        (EDITED_ARTICLE, _('edited')),
        (ALSO_COMMENTED, _('also commented')),
        (LOGGED_IN, _('logged in')),
        (LOGGED_OUT, _('logged out')),
        (VOTED, _('voted on')),
        (SHARED, _('shared')),
        (SIGNUP, _('created an account')),
        (REPLY, _('replied to'))
        )
    actor = models.ForeignKey(settings.AUTH_USER_MODEL,
                              related_name="notify_actor",
                              on_delete=models.CASCADE)
    recipient = models.ForeignKey(settings.AUTH_USER_MODEL, blank=False,
        related_name="notifications", on_delete=models.CASCADE)
    unread = models.BooleanField(default=True, db_index=True)
    timestamp = models.DateTimeField(auto_now_add=True)
    uuid_id = models.UUIDField(
        primary_key=True, default=uuid.uuid4, editable=False)
    slug = models.SlugField(max_length=210, null=True, blank=True)
    verb = models.CharField(max_length=1, choices=NOTIFICATION_TYPES)
    action_object_content_type = models.ForeignKey(ContentType,
        blank=True, null=True, related_name="notify_action_object",
        on_delete=models.CASCADE)
    action_object_object_id = models.CharField(
        max_length=50, blank=True, null=True)
    action_object = GenericForeignKey(
        "action_object_content_type", "action_object_object_id")
    objects = NotificationQuerySet.as_manager()

    class Meta:
        verbose_name = _("Notification")
        verbose_name_plural = _("Notifications")
        ordering = ("-timestamp",)

    def __str__(self):
        if self.action_object:
            return f'{self.actor} {self.get_verb_display()}
{self.action_object} {self.time_since()} ago'

        return f'{self.actor} {self.get_verb_display()} {self.time_since()}
ago'

    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(f'{self.recipient} {self.uuid_id}
{self.verb}',
                                to_lower=True, max_length=200)

        super().save(*args, **kwargs)

    def time_since(self, now=None):
        """
        Shortcut for the ``django.utils.timesince.timesince`` function of
the
        current timestamp.
        """
        from django.utils.timesince import timesince

        return timesince(self.timestamp, now)

    def get_icon(self):
        """Model method to validate notification type and return the closest
        icon to the verb.
        """
        if self.verb == 'C' or self.verb == 'A' or self.verb == 'K':
            return 'fa-comment'

        elif self.verb == 'I' or self.verb == 'U' or self.verb == 'O':
            return 'fa-users'

        elif self.verb == 'L':
            return 'fa-heart'

        elif self.verb == 'F':
            return 'fa-star'

        elif self.verb == 'W':
            return 'fa-check-circle'

        elif self.verb == 'E':
            return 'fa-pencil'

        elif self.verb == 'V':
            return 'fa-plus'

        elif self.verb == 'S':
            return 'fa-share-alt'

        elif self.verb == 'R':
            return 'fa-reply'

    def mark_as_read(self):
        if self.unread:
            self.unread = False
            self.save()

    def mark_as_unread(self):
        if not self.unread:
            self.unread = True
            self.save()


def notification_handler(actor, recipient, verb, **kwargs):
    """
    Handler function to create a Notification instance.
    :requires:
    :param actor: User instance of that user who makes the action.
    :param recipient: User instance, a list of User instances or string
                      'global' defining who should be notified.
    :param verb: Notification attribute with the right choice from the list.

    :optional:
    :param action_object: Model instance on which the verb was executed.
    :param key: String defining what kind of notification is going to be
created.
    :param id_value: UUID value assigned to a specific element in the DOM.
    """
    key = kwargs.pop('key', 'notification')
    id_value = kwargs.pop('id_value', None)
    if recipient == 'global':
        users =
get_user_model().objects.all().exclude(username=actor.username)
        for user in users:
            Notification.objects.create(
                actor=actor,
                recipient=user,
                verb=verb,
                action_object=kwargs.pop('action_object', None)
            )
        notification_broadcast(actor, key)

    elif isinstance(recipient, list):
        for user in recipient:
            Notification.objects.create(
                actor=actor,
                recipient=get_user_model().objects.get(username=user),
                verb=verb,
                action_object=kwargs.pop('action_object', None)
            )

    elif isinstance(recipient, get_user_model()):
        Notification.objects.create(
            actor=actor,
            recipient=recipient,
            verb=verb,
            action_object=kwargs.pop('action_object', None)
        )
        notification_broadcast(
            actor, key, id_value=id_value, recipient=recipient.username)

    else:
        pass


def notification_broadcast(actor, key, **kwargs):
    """Notification handler to broadcast calls to the recieve layer of the
    WebSocket consumer of this app.
    :requires:
    :param actor: User instance of that user who makes the action.
    :param key: String parameter to indicate the client which action to
                perform.

    :optional:
    :param id_value: UUID value assigned to a specific element in the DOM.
    :param recipient: String indicating the name of that who needs to be
                      notified.
    """
    channel_layer = get_channel_layer()
    id_value = kwargs.pop('id_value', None)
    recipient = kwargs.pop('recipient', None)
    payload = {
            'type': 'receive',
            'key': key,
            'actor_name': actor.username,
            'id_value': id_value,
            'recipient': recipient
        }
    async_to_sync(channel_layer.group_send)('notifications', payload)
```

Table 5.6 Notification models.py

```
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.shortcuts import get_object_or_404, redirect, render
from django.utils.translation import ugettext_lazy as _
from django.views.generic import ListView

from onlinemechaniclocator.notifications.models import Notification


class NotificationUnreadListView(LoginRequiredMixin, ListView):
    """Basic ListView implementation to show the unread notifications for
    the actual user"""
    model = Notification
    context_object_name = 'notification_list'
    template_name = 'notifications/notification_list.html'

    def get_queryset(self, **kwargs):
        return self.request.user.notifications.unread()


@login_required
def mark_all_as_read(request):
    """View to call the model method which marks as read all the
notifications
    directed to the actual user."""
    request.user.notifications.mark_all_as_read()
    _next = request.GET.get('next')
    messages.add_message(
        request, messages.SUCCESS,
        _(f'All notifications to {request.user.username} have been marked as
read.'))

    if _next:
        return redirect(_next)

    return redirect('notifications:unread')


@login_required
def mark_as_read(request, slug=None):
    """View to call the model method which mark as read the provided
    notification."""
    if slug:
        notification = get_object_or_404(Notification, slug=slug)
        notification.mark_as_read()

    messages.add_message(
        request, messages.SUCCESS,
        _(f'The notification {notification.slug} has been marked as read.'))
    _next = request.GET.get('next')

    if _next:
        return redirect(_next)

    return redirect('notifications:unread')


@login_required
```

```
def get_latest_notifications(request):
    notifications = request.user.notifications.get_most_recent()
    return render(request,
                  'notifications/most_recent.html',
                  {'notifications': notifications})
```

Table 5.7 Notification views.py

```
from django.conf.urls import url

from onlinemechaniclocator.notifications import views

app_name = 'notifications'
urlpatterns = [
    url(r'^$', views.NotificationUnreadListView.as_view(), name='unread'),
    url(r'^mark-as-read/(?P<slug>[-\w]+)/$', views.mark_as_read,
name='mark_as_read'),
    url(r'^mark-all-as-read/$', views.mark_all_as_read,
name='mark_all_read'),
    url(r'^latest-notifications/$', views.get_latest_notifications,
name='latest_notifications'),
]
```

Table 5.8 Notifications url.py

5.2.1 Code Efficiency

Code efficiency is a broad term used to depict the reliability, speed and programming methodology used in developing codes for an application. Code efficiency is directly linked with algorithmic efficiency and the speed of runtime execution for software. It is the key element in ensuring high performance. The goal of code efficiency is to reduce resource consumption and completion time as much as possible with minimum risk to the business or operating environment. The software product quality can be accessed and evaluated with the help of the efficiency of the code used.

5.3 Testing Approaches

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be tested in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model. Various test cases were described in Section 4.6. In this section, we will test all the scenarios discussed and check whether the outcome that we receive is the expected outcome that we predicted earlier.

5.3.1 Unit Testing

ID	Name	Description	Steps	Expected outcome
TC1	Login test case	To test the Functionality Of the user	4. Login with the Registered user in the Database 5. Login with partially Correct information 6. Login with no information.	The application successfully Logins the user when Both email and password Are correct otherwise Display appropriate Error message to the user
TC2	Reset Password Test case	To test Weather user can reset password if the user forgets it	5. Enter email address and the application should click on reset password, be able to send reset 6. Open user email and password email to click on specified link. The user and update 7. Input the new password. The password upon 8. Login with the new password rest. Password.	The application is able to send reset password email to the user and update the password up on password reset.

TC3	Change Password Test case	To test whether user can change password if user wishes to change	5. Login using the user credentials. 6. Open user profile and open change password dialog. 7. Input current password and new password twice. 8. Logout and try to login with new password	The application is able to change user's password if the user knows it's current password, the same should reflect when user tries to login next time.
TC4	Broadcasting Question Problem Test case	To test whether the user can broadcast a problem question	4. Login using the user credentials. 5. In home page, select QA option and type the problem question in the text field. 6. Click on Send button to broadcast it.	The application is able to broadcast problem questions which user inputs.
TC5	Logout test case	To test the functionality of logging out of the user	3. Click on the Gear icon 4. Scroll & Click on the logout button.	The application is able to logout the current user, and display login screen.

Table 5.9 Test Cases

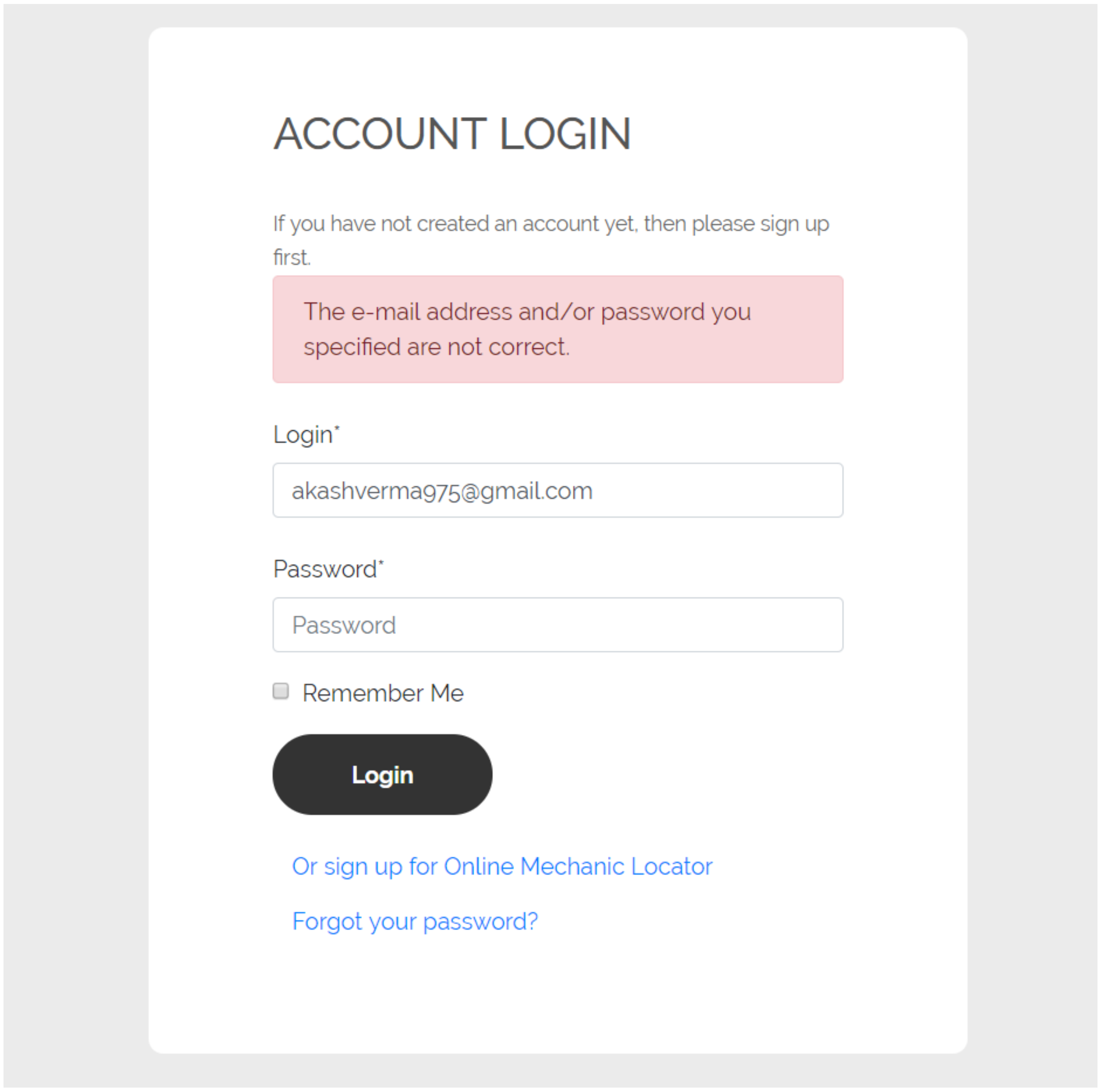
5.3.2 Integrated Testing

Integrated Testing brings all the modules together into a special testing environment, then checks for errors, bugs and interoperability. Basically, the Test Case TCR04 deals with the integration testing of the system as a whole. It deals with tests for the entire application. Application limits and features are tested here. ID Name Description TCR04 Integrated Testing to test the functionality of all the components together.

CHAPTER 6: RESULTS AND DISCUSSION

6.1 Test Reports

TC1 is the Login test case which tests the functionality of the website to successfully log in the student when correct information is given, otherwise display appropriate message.



ACCOUNT LOGIN

If you have not created an account yet, then please sign up first.

The e-mail address and/or password you specified are not correct.

Login*

akashverma975@gmail.com

Password*

Password

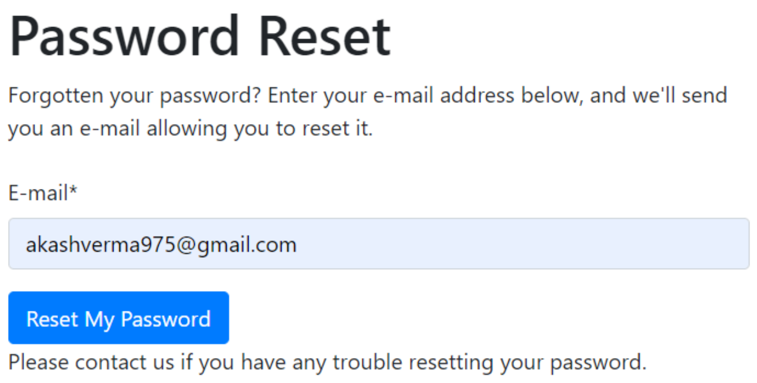
Remember Me

Login

Or sign up for Online Mechanic Locator

Forgot your password?

Figure 6.1 Test Report of the Login Test Case – Case 1



Password Reset

Forgotten your password? Enter your e-mail address below, and we'll send you an e-mail allowing you to reset it.

E-mail*

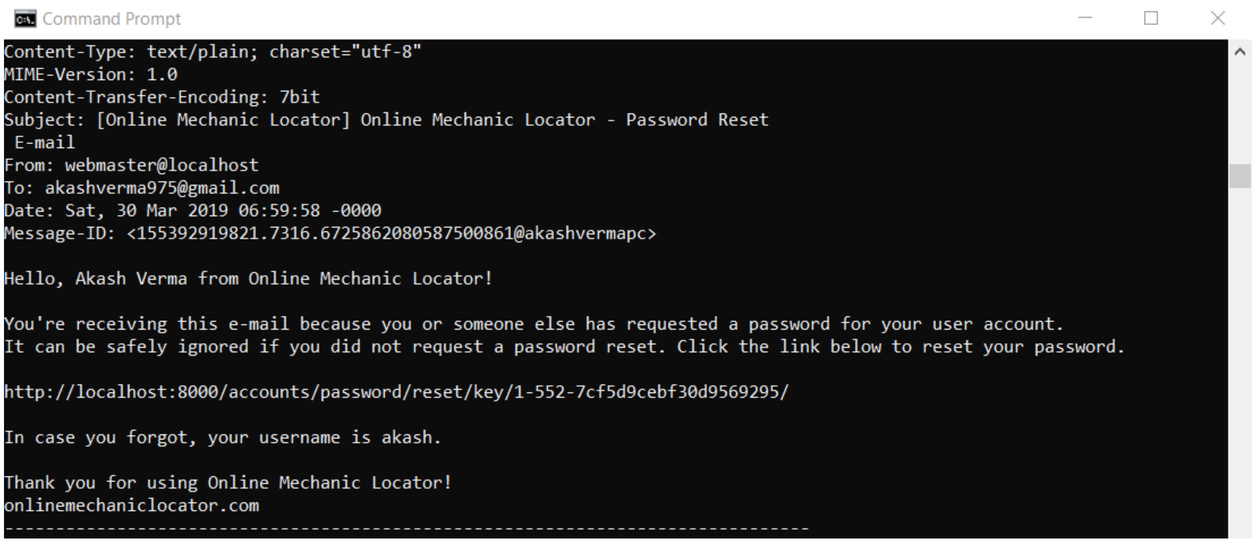
akashverma975@gmail.com

Please contact us if you have any trouble resetting your password.

Figure 6.2 Test Report of the Password Reset Test Case – Case 1

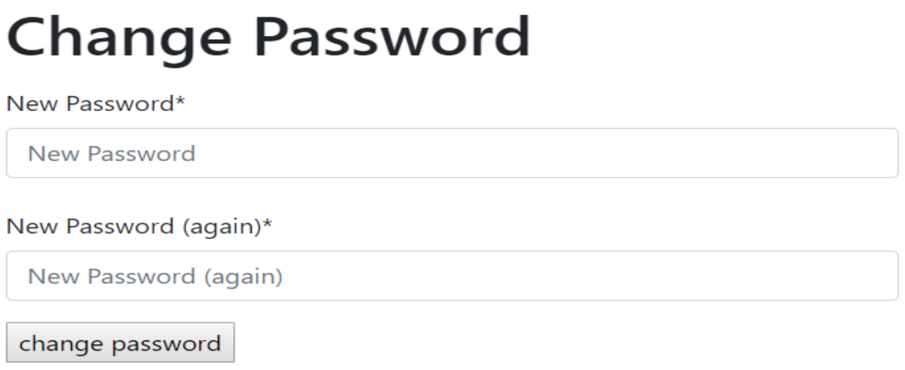
Case 1 occurs when there is an unsuccessful login, whereas Case 2 is the landing page when the login is successful.

TC2 is the Reset Password test case which tests the functionality of the website to reset the password of the student through email verification.



```
Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Subject: [Online Mechanic Locator] Online Mechanic Locator - Password Reset
 E-mail
From: webmaster@localhost
To: akashverma975@gmail.com
Date: Sat, 30 Mar 2019 06:59:58 -0000
Message-ID: <155392919821.7316.6725862080587500861@akashvermapc>

Hello, Akash Verma from Online Mechanic Locator!

You're receiving this e-mail because you or someone else has requested a password for your user account.
It can be safely ignored if you did not request a password reset. Click the link below to reset your password.

http://localhost:8000/accounts/password/reset/key/1-552-7cf5d9cebf30d9569295/

In case you forgot, your username is akash.

Thank you for using Online Mechanic Locator!
onlinemechaniclocator.com
-------------------------------------------------------------------------------
```

Figure 6.3 Test Report of the Password Reset Test Case – Case 2


Change Password

New Password*

New Password

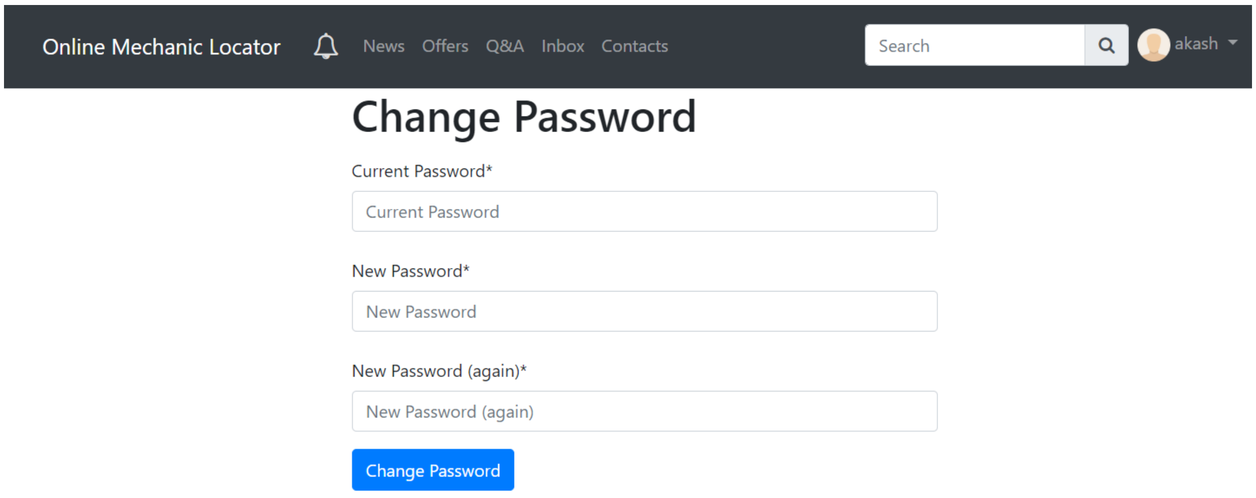
New Password (again)*

New Password (again)

Figure 6.4 Test Report of the Password Reset Test Case – Case 3

In the fig above, an email is sent to the user if the reset password option is chosen at the login page. Upon clicking the link, the user is redirected back to the website and is prompted to enter a new password.

TC3 is the Change Password test case which tests the functionality of the website to change the password of the logged in user by verifying the current password of the user.



Change Password

Current Password*

Current Password

New Password*

New Password

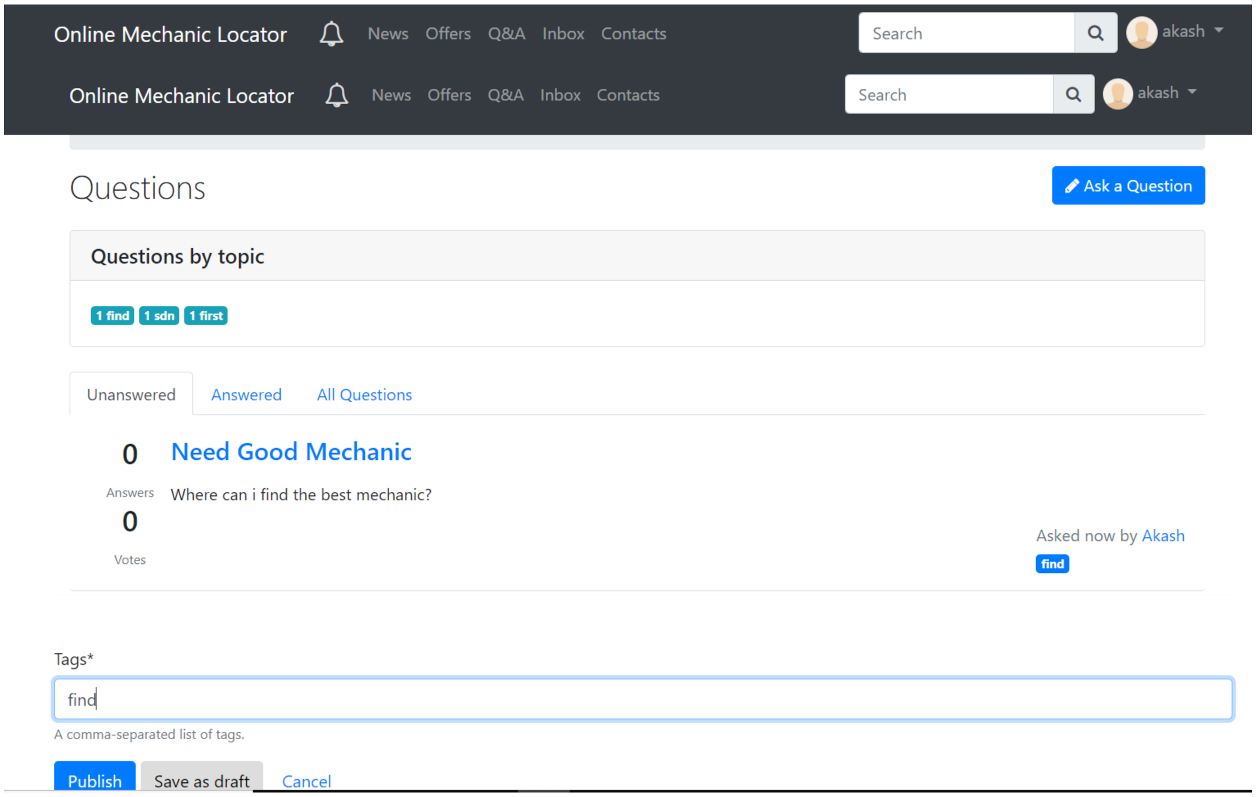
New Password (again)*

New Password (again)

Figure 6.5 Test Report of the Password Reset Test Case – Case 3.1

In the above figure, when the user clicks on the change password option. The password change page then asks to enter the current password, and then the new password. It validates the current password before actually changing the password.

TC4 is Broadcasting Question Problem Test case which test whether user is able to broadcast its question to all the mechanics and other user in the system.



Questions

Questions by topic

1 find 1 sdn 1 first

0 Answers 0 Votes

Need Good Mechanic

Where can i find the best mechanic?

Asked now by Akash

find

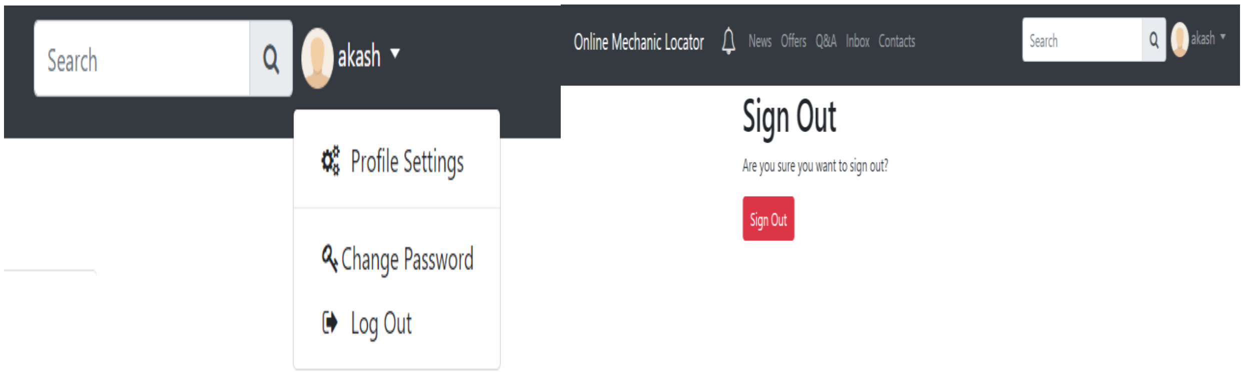
Tags*

find

A comma-separated list of tags.

Figure 6.6 Test Report of the Questions Test Case – Case 4

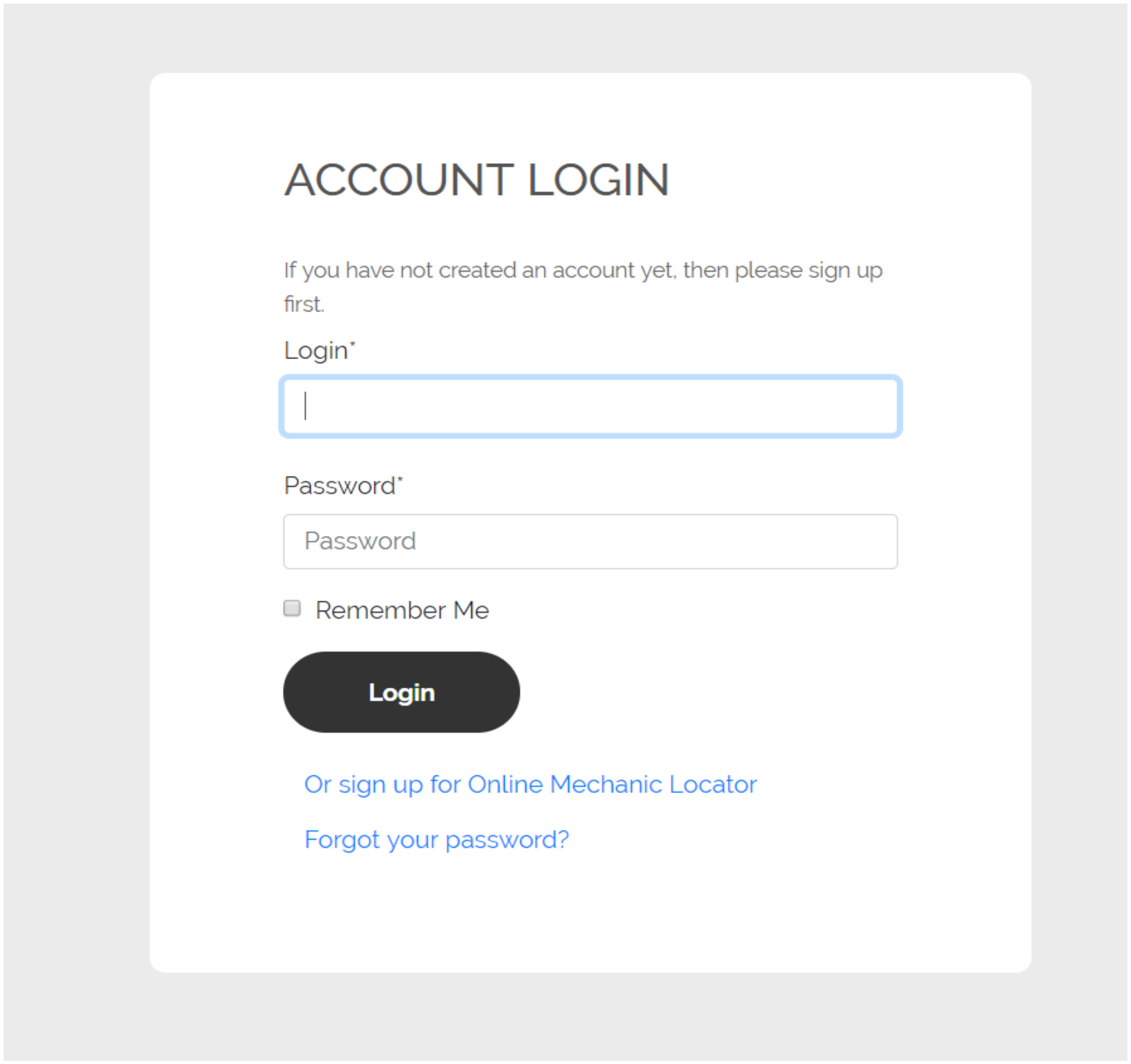
TC5 is the Logout test case which tests the functionality of the website to successfully logout the currently logged in user and display the login screen again.



Sign Out

Are you sure you want to sign out?

Figure 6.7 Test Report of the Sign Out Test Case – Case 5


ACCOUNT LOGIN

If you have not created an account yet, then please sign up first.

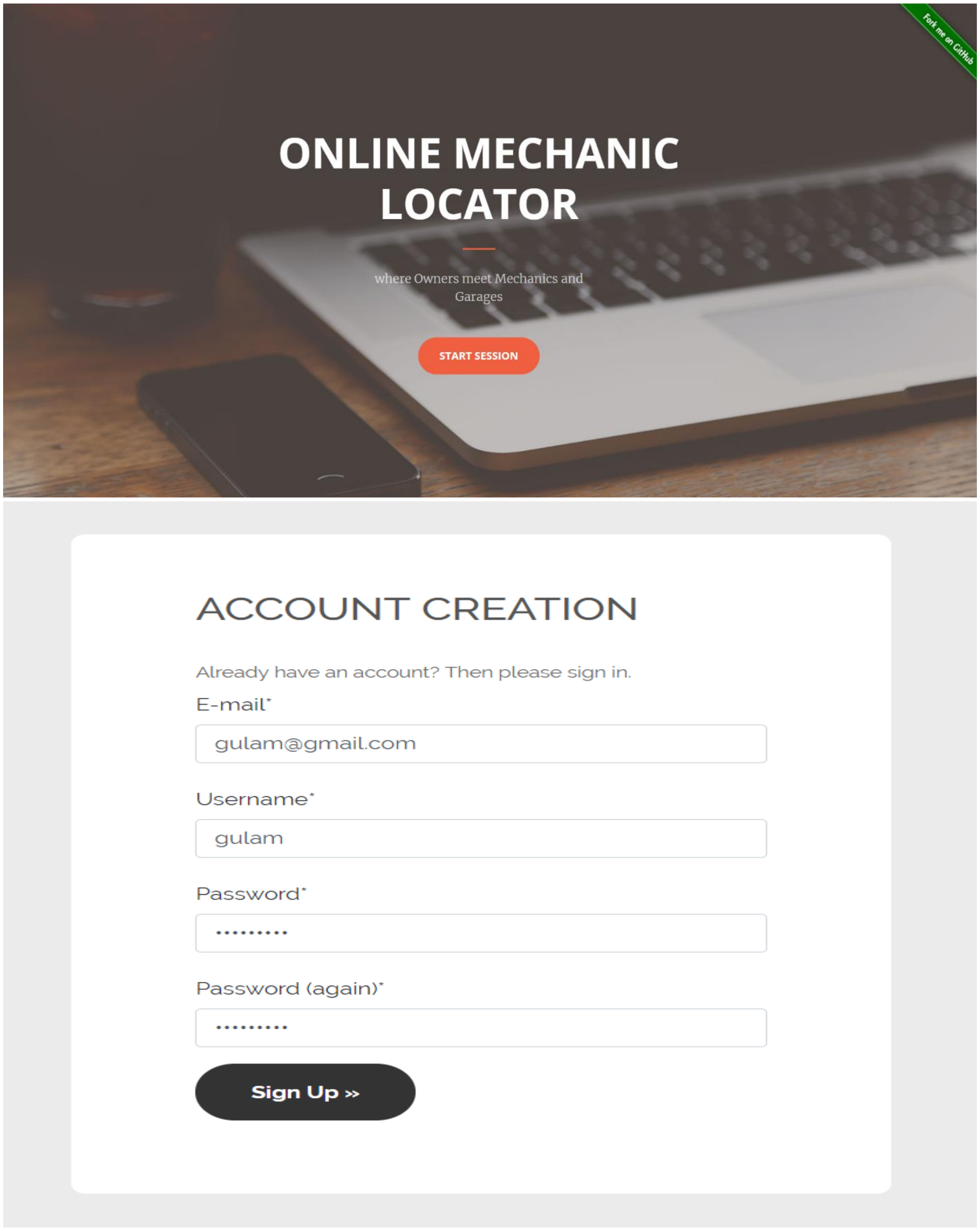
Login*

Password*

Figure 6.8 Test Report of the Sign Out Test Case – Case 5.1

6.2 User Documentation

6.2.1 User Registration



ACCOUNT CREATION

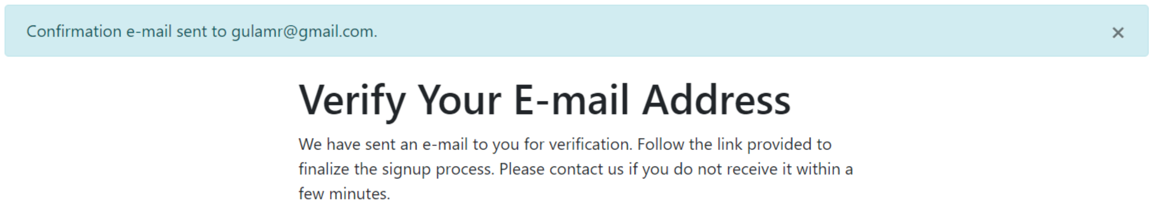
Already have an account? Then please sign in.

E-mail*
gulam@gmail.com

Username*
gulam

Password*
•••••••••

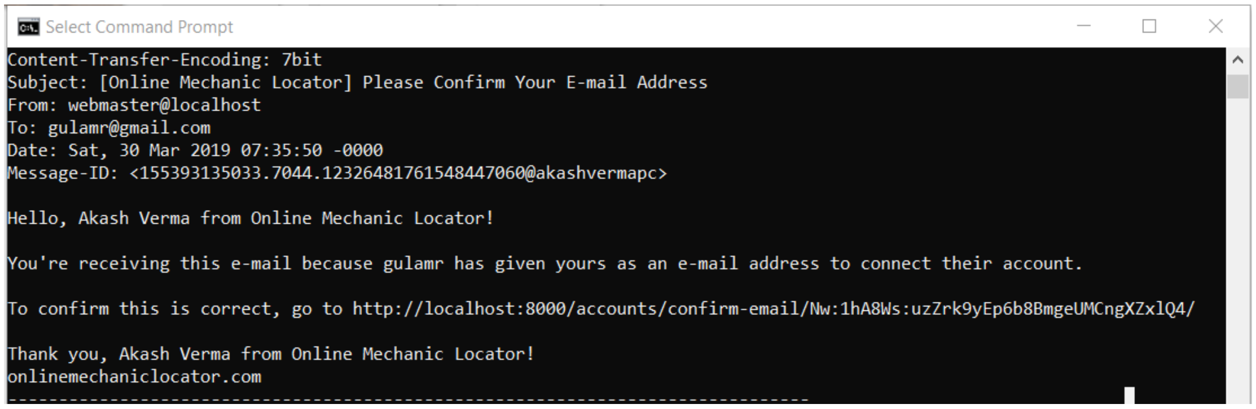
Password (again)*
•••••••••



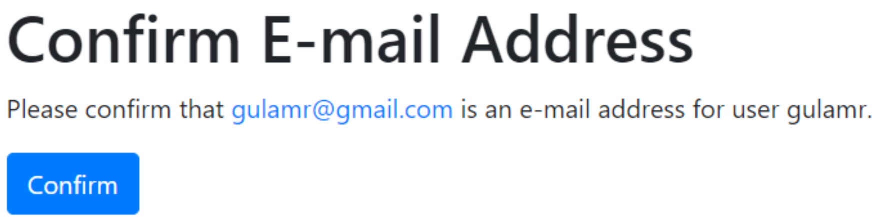
Confirmation e-mail sent to gulamr@gmail.com.

Verify Your E-mail Address

We have sent an e-mail to you for verification. Follow the link provided to finalize the signup process. Please contact us if you do not receive it within a few minutes.



```
Content-Transfer-Encoding: 7bit
Subject: [Online Mechanic Locator] Please Confirm Your E-mail Address
From: webmaster@localhost
To: gulamr@gmail.com
Date: Sat, 30 Mar 2019 07:35:50 -0000
Message-ID: <155393135033.7044.12326481761548447060@akashvermapc>

Hello, Akash Verma from Online Mechanic Locator!

You're receiving this e-mail because gulamr has given yours as an e-mail address to connect their account.

To confirm this is correct, go to http://localhost:8000/accounts/confirm-email/Nw:1hA8Ws:uzZrk9yEp6b8BmgeUMCngXZxlQ4/

Thank you, Akash Verma from Online Mechanic Locator!
onlinemechaniclocator.com
-------------------------------------------------------------------------------
```

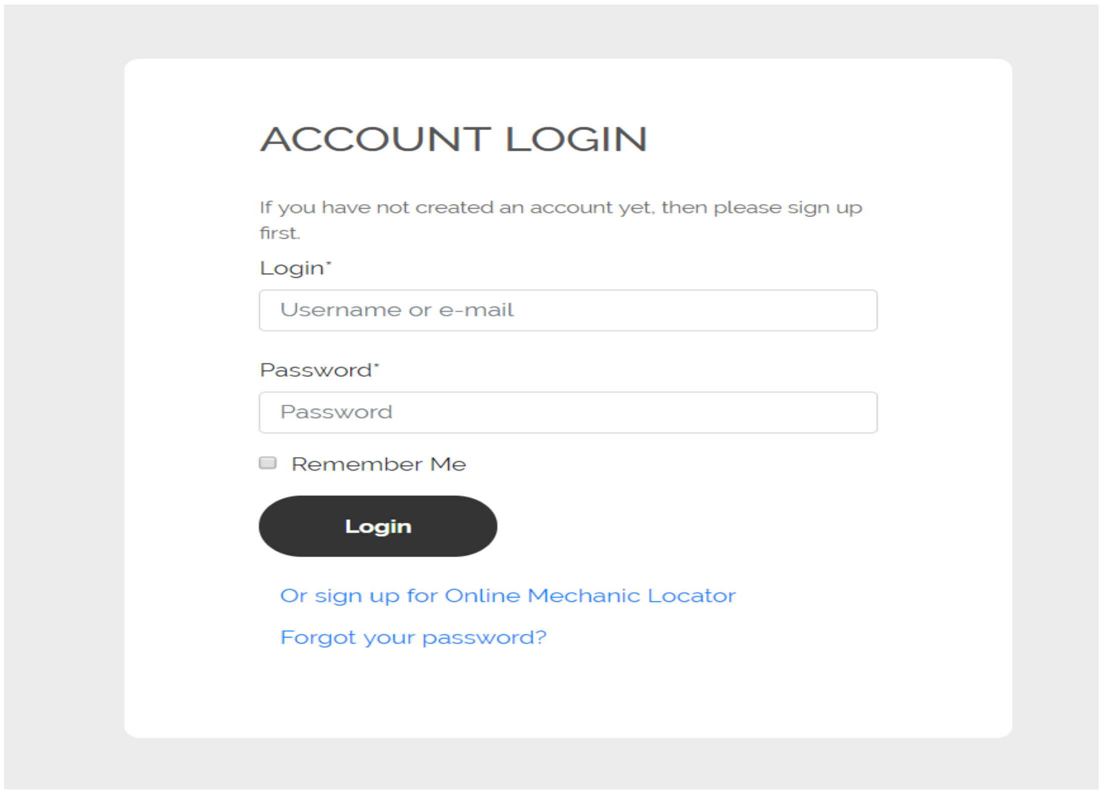


Confirm E-mail Address

Please confirm that gulamr@gmail.com is an e-mail address for user gulamr.

Figure 6.9 User Registration

6.2.2 User Login and Interactions



ACCOUNT LOGIN

If you have not created an account yet, then please sign up first.

Login*

Username or e-mail

Password*

Password

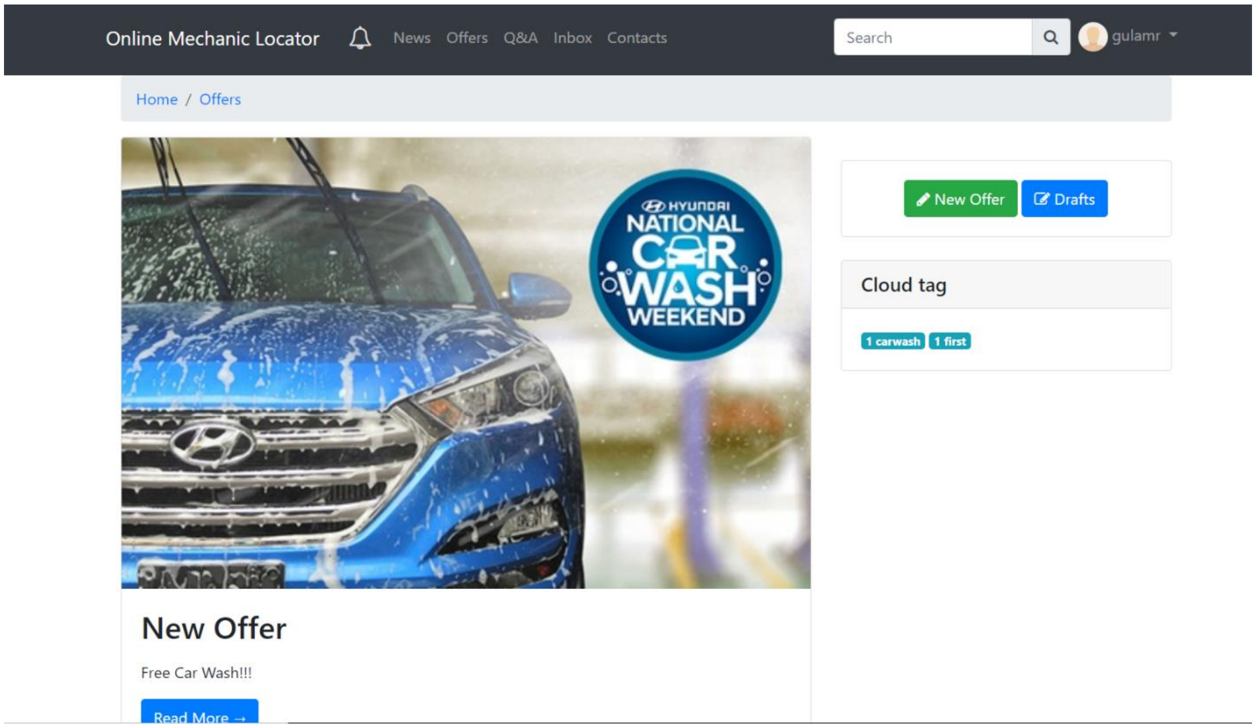
Remember Me

Login

Or sign up for Online Mechanic Locator

Forgot your password?

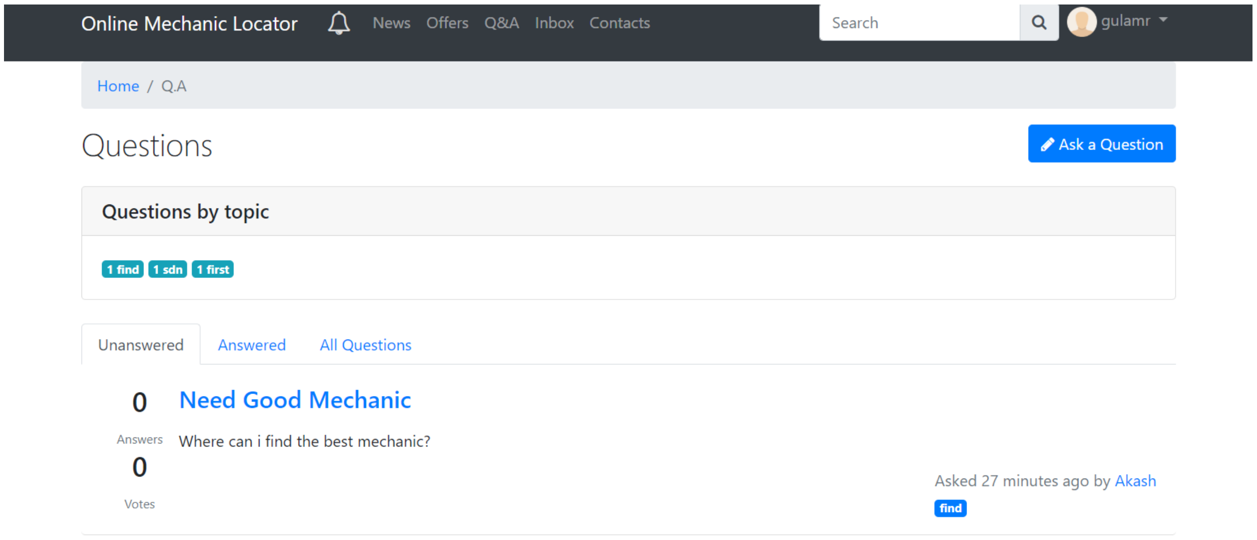
Figure 6.10 Login Page



New Offer

Free Car Wash!!!

Figure 6.11 Home Page


Online Mechanic Locator News Offers Q&A Inbox Contacts Search gulamr

Home / Q.A

Questions

Ask a Question

Questions by topic

1 find 1 sdn 1 first

Unanswered Answered All Questions

0 Answers 0 Votes

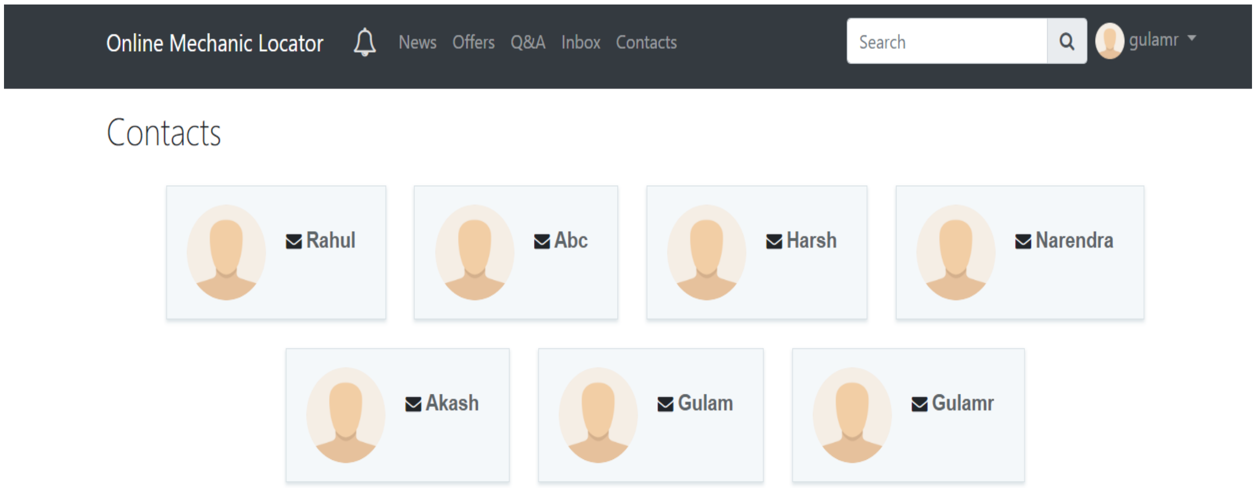
Need Good Mechanic

Where can i find the best mechanic?

Asked 27 minutes ago by Akash

find

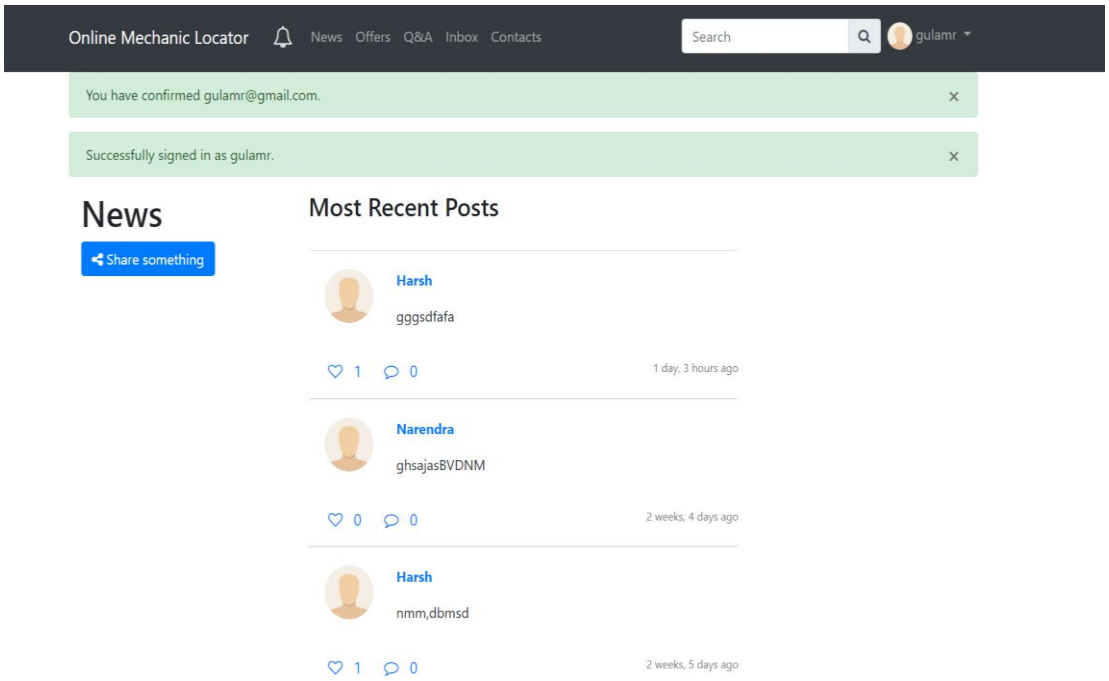
Figure 6.12 Questions Page


Online Mechanic Locator News Offers Q&A Inbox Contacts Search gulamr

Contacts

Rahul Abc Harsh Narendra Akash Gulam Gulamr

Figure 6.13 Contacts Page



You have confirmed gulamr@gmail.com.

Successfully signed in as gulamr.

News

Most Recent Posts

Harsh

gggsdfafa

1 0 1 day, 3 hours ago

Narendra

ghsajasBVDNM

0 0 2 weeks, 4 days ago

Harsh

nmm,dbmsd

1 0 2 weeks, 5 days ago

Figure 6.14 Posts Page

CHAPTER 7: CONCLUSION

7.1 Conclusion

Online mechanic Locator help user to connect with mechanic and garages nearby. The users can also share amazing offers and news among themselves. The website also has the feature of QnA where the users can ask the question to the mechanics and other users and they can also upvote or down-vote the question asked to avoid spam. Messaging system in the website can help user to chat privately with mechanics and garages ensuring personalized experience on the website. This project has helped in understanding the working of how a project is implemented and how problems are solved. From ideation state to testing the component and integrating it with the system, I have understood the importance of following the software development model while developing the project. It has given me the opportunity to explore myself and let myself go out of my comfort zone. I worked with the technologies like Git and GitHub and they have proved their significance in product development and why they are always recommended while working with software projects. I have worked with Django in this project and I now know the significance of it and why it is been used by websites like Instagram and NASA.

7.2 Limitations of the System

No system in this world is perfect. The current limitation of the system is that is can be daunting task to actually find the mechanic on map just my looking at their address from their profile.

The second limitation is that the system is not been tested in real working environments with the real traffic and real people.

While working with this project I have come across a numerous solution to the problems faced by developers while developing an application which is scalable, reliable, secured and maintainable. Few of the solutions are mentioned below:-

Containerization – The development environment and deployment environment are very different. It may happen that something works on development environment but it doesn't work on the deployment environment and vice-verse. To solve this problem we can make use of Virtualization software and create a deployment environment in our local machine and work on it but again the problem of resource utilization come here as virtualization software runs on host OS. The solution to this problem is Containerization.

Containerization is a lightweight alternative to full machine virtualization that involves encapsulating an application in a container with its own operating environment. This provides many of the benefits of loading an application onto a virtual machine, as the application can be run on any suitable physical machine without any worries about dependencies.

Microservices are a software development technique—a variant of the service-oriented architecture (SOA) architectural style that structures an application as a collection of loosely coupled services. In a microservices architecture, services are finegrained and the protocols are lightweight. The benefit of decomposing an application into different smaller services is that it improves modularity. This makes the application easier to understand, develop, test, and become more resilient to architecture erosion. It parallelizes development by enabling small autonomous teams to develop, deploy and scale their respective services independently. It also allows the architecture of an individual service to emerge through continuous refactoring. Microservices-based architectures enable continuous delivery and deployment.

7.3 Future Scope of the Project

Map based location tracking and live traffic updates can be implemented in future to make the application more usable. Android and IOS app can be made to provide an accurate GPS location of the mechanic to user. Online purchasing of parts and Online Payment to mechanic services can be implemented making sure that the application is secured technically and morally.

References

Stack OverFlow - https://stackoverflow.com/

Draw.io - https://www.draw.io/

W3Schools - https://www.w3schools.com/

Django Projects - https://www.djangoproject.com/

Django Documentations - https://docs.djangoproject.com/en/2.1/

Pillow - https://github.com/python-pillow/Pillow

Psycopg2 - https://github.com/psycopg/psycopg2

Redis - https://github.com/antirez/redis

Whitenoise - https://github.com/evansd/whitenoise

Pythin Decouple - https://github.com/henriquebastos/python-decouple

Django Allauth - https://www.djangoproject.comhttps//github.com/pennersr/django-allauth

Django Crispy Forms - https://github.com/django-crispy-forms/django-crispy-forms

Django Environ - https://github.com/joke2k/django-environ

Django Redis - https://github.com/niwinz/django-redis

Django Channels - https://github.com/django/channels

Django Channels Redis - https://github.com/django/channels_redis

Django Extensions - https://github.com/django-extensions/django-extensions