EMERGENO-EMERGENCY SERVICE MANAGEMENT SYSTEM

Project Report SubmittedBy

ALANTINA MATHEW

Reg. No.: AJC17MCA-I004

In Partial fulfillment for the Award of the Degree Of

INTEGRATED MASTER OF COMPUTER APPLICATIONS (INMCA)
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

2017-2022

DEPARTMENT OF COMPUTER APPLICATIONS

AMAL JYOTHI COLLEGE OF ENGINEERING

KANJIRAPPALLY

<u>CERTIFICATE</u>

This is to certify that the Project report, "**EMERGENCY SERVICE MANAGEMENT"** is the bonafide work of **ALANTINA MATHEW (Reg.No:AJC17MCA-I004)** in partial fulfillment of the requirements for the award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2021-22.

Ms. Lisha Varghese	**Rev.Fr.Dr.Rubin Thottupurathu Jose**
Internal Guide	**Coordinator**

Rev.Fr.Dr.Rubin Thottupurathu Jose	**External External Examiner**
Head of the Department	

DECLARATION

I hereby declare that the project report **"EMERGENCY SERVICE MANAGEMENT SYSTEM"** is a bonafide work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Degree of Integrated Master of Computer Applications (MCA) from APJ Abdul Kalam Technological University, during the academic year 2021-2022.

Date: 22/05/2022 **ALANTINA MATHEW**

KANJIRAPPALLY **Reg. No: AJC17MCA-I004**

ACKNOWLEDGEMENT

First and foremost, I thank God Almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jacob** for providing good faculty guidance.

I owe a great depth of gratitude towards our Head of the Department and project coordinator **Rev. Fr. Dr. Rubin Thottupurathu Jose** for their valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also like to express sincere gratitude to my guide**, Ms**. **Lisha Varghese** for her inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

ALANTINA MATHEW

ABSTRACT

Emergency Assistant is an online platform that helps to give adequate services for emergency needs. The purpose of the system is to develop an integrated emergency service management system that is easily accessible to the public. This system will be a solution for all emergency needs like blood requirements, ambulance service, delivery of fuel, mechanic service. The main idea is to implement an automated software application for maintaining the services for emergency needs. The required software and hardware are easily available.

Emergency Assistant is helpful for people who want an emergency service. In this software application, People can search for blood donors near them, People can request nearby ambulance services, fuel agents, and mechanics and can share their location to them. This system is more secure, reliable, and fast. For emergency service providers, giving their service in least time shows their best performance. To assuring that we are using current location share. This will help in better utilization of resources.

.

CONTENT

List of Abbreviation

IDE	-	Integrated Development Environment
HTML	-	Hyper Text Markup Language.
CSS	-	Cascading Style Sheet
SQL	-	Structured Query Language
UML	-	Unified Modeling Language

CHAPTER 1

INTRODUCTION

1.1 PROJECT OVERVIEW

“EMERGENCY SERVICE MANAGEMENT SYSTEM” is a web application that is meant to help customers in emergencies. The customer can also reduce the time and effort in searching different emergency applications by using this system. This system will be a solution for all emergency needs like blood requirements, ambulance calls, fuel, vehicle repair, etc.

The proposed system includes five users they are administrator, customer, ambulance driver, fuel attendant (fuelbuddy), and mechanic. The customer can log in to the website and request the emergency service when they need. This project aims at offering the best emergency services to clients in need. It is an automated software application with minimum cost. This system is more secure, reliable, and fast. This will help in better utilization of resources.

1.2 PROJECT SPECIFICATION

Emergency Assistant is helpful for the people who require an emergency service of nearby blood donors, ambulance service, Fuel Buddy and GoMechanic. This system is a web-based application.

It has 5 Modules. They are:-

1. **Admin Module**

   Admin must have a login into this system. He has overall control of the system. Admin can View all the registered users and also manage all his data. Admin can also view the requests given and services taken by the users.

2. **Customer Module**

   Customers can register and they can search nearby blood donors, ambulance services, Fuel Buddy and GoMechanic. They can request these emergency services when they need them.

3. **Blood Donor Module**

   Customers can also be blood donors by registering them as a blood donors. They can give their name and phone number so that customers can find them easily.

4. **Ambulance Driver Module**

   Ambulance drivers can register on the website. They can provide a Customer Care Number so the customers can easily find out. They can also mention the cities in which they are providing services.

5. **GoMechanic Module**

   Mechanics can register their shop on the website. They can mention the services provided by them on this site. They can also mention their customer care number. Customers can do their payments either online or offline

6. **Fuel buddy Module**

   Fuel Buddy helps the customer to get fuel on the doorstep. Various fuel agencies can register on the website and can provide their service. They can mention the customer care number, fuel prices on the website

CHAPTER 2
SYSTEM STUDY

2.1 INTRODUCTION

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem-solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minute's detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system is identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors, and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive data conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is a loop that ends as soon as the user is satisfied with the proposal.

A preliminary study is a process of gathering and interpreting facts, using the information for further studies on the system. The preliminary study is a problem-solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

2.2 EXISTING SYSTEM

The existing system is not a fully automated system. In the existing system, we have to depend on multiple software applications from different vendors for availing of emergency services which will make the user uncomfortable. Many of the services are working manually. It's more time-consuming.

2.3 DRAWBACK OF EXISTING SYSTEM

- High Cost
- Huge Size
- Complexity
- Need to download multiple apps
- Information Sharing is not possible
- Time Consuming

2.4 PROPOSED SYSTEM

The proposed system tries to eliminate or reduce difficulties up to some extent. This will help to reduce the time consumed and the effort. This system acts as an integrated system for all emergency services. It provides to management timely, confidential, and secure services that facilitate planning and decision making. It provides more security to the user via sharing live locations. It gives an instant solution to emergency problems such as blood donors, ambulance services, and vehicle breakdowns.

2.5 ADVANTAGES OF THE PROPOSED SYSTEM

- User-friendliness and interaction.
- Minimum time required.
- Security of data.
- Minimum time needed for the various processing.
- Greater efficiency.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDY

The feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort, and time that spend on it. A feasibility study lets the developer foresee the future of the project and its usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs, and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

3.1.1 Technical Feasibility

Technical feasibility is concerned with the availability of hardware and software required for the development of the system, to see compatibility and maturity of the technology proposed to be used, and to see the availability of the required technical manpower to develop the system. After the study, we came to the conclusion that we proceed further with the tools and development environment chosen by us. This was important in our case as we were working on two various phases of the department that will need to be integrated in the future to make an extended system. So, it's clear that this project is technically feasible.

3.1.2 Resource feasibility

This aspect looks at the resources that are required to complete the project and whether the amount of available resources is sufficient to complete the project effectively. Resources that are required for this project includes Programming device (Laptop), Hosting space (freely available), Programming tools (freely available), Programming individuals. So, it's clear that this project has the required resource feasibility.

3.1.3 Economic Feasibility

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide. Being a web application will have an associated hosting cost. Since the system doesn't consist of any multimedia data transfer, the bandwidth required for the operation of this application is very low. Bug fixes and maintaining tasks will have an associated cost. Besides the associated cost, there will be many benefits for the customers. Especially the extra effort that is associated with papermaking and marking will be significantly reduced while the effort to create descriptive statistical reports will be eliminated. From there, it's clear that the project is Economic feasible.

3.2 SYSTEM SPECIFICATION

3.2.1 HARDWARE REQUIREMENTS:

Processor	: Intel core i3, or above
RAM	: 4 GB or above
Mouse	: Standard Mouse
Keyboard	: Logitech Keyboard
Processor Speed	: 2.20 GHz

3.2.2 SOFTWARE REQUIREMENTS:

Operating System	: Windows 8.1 or later
Front End Design	: HTML5, CSS3, Bootstrap
Front End Development	: JavaScript, JQuery, Ajax
Back End Development	: Laravel
Database	: MySql
Web Server	: XAMPP
IDE	: Visual Studio Code

3.3 SOFTWARE DESCRIPTION

3.3.1 Laravel

Laravel is an open-source PHP framework, which is robust and easy to understand. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application. The web application thus designed is more structured and pragmatic.

Laravel offers a rich set of functionalities which incorporates the basic features of PHP frameworks like CodeIgniter, Yii and other programming languages like Ruby on Rails. Laravel has a very rich set of features which will boost the speed of web development.

If you are familiar with Core PHP and Advanced PHP, Laravel will make your task easier. It saves a lot time if you are planning to develop a website from scratch. Moreover, a website built in Laravel is secure and prevents several web attacks.

3.3.2 PHP

The PHP Hypertext Preprocessor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications. PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994. PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.

It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server. PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the Unix side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time. PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time. PHP is forgiving: PHP language tries to be as forgiving as possible. PHP Syntax is C-Like.

3.3.3 MySQL

MySQL5is a relational database management system based on the Structured Query Language, which is the popular language for accessing and managing the records in the database. MySQL is open-source and free software under the GNU license. It is supported by Oracle Company.

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with PHP scripts for creating powerful and dynamic server-side or web-based enterprise applications.

MySQL is Relational Database Management System (RDBMS) software that provides many things, which are as follows:

It allows us to implement database operations on tables, rows, columns, and indexes.

It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.

It provides the Referential Integrity between rows or columns of various tables. It allows us to updates the table indexes automatically.

It uses many SQL queries and combines useful information from multiple tables for the end-users.

MySQL is becoming so popular because of these following reasons:

MySQL is an open-source database, so you don't have to pay a single penny to use it. MySQL is a very powerful program that can handle a large set of functionality of the most expensive and powerful database packages.

MySQL is customizable because it is an open-source database, and the open-source GPL license facilitates programmers to modify the SQL software according to their own specific environment.

CHAPTER 4

SYSTEM DESIGN

4.1 INTRODUCTION

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML stands for Unified Modelling Language. UML is a pictorial language used to make software blueprints. UML can be described as a general-purpose visual modelling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object-oriented analysis and design

4.2.1 USE CASE DIAGRAM

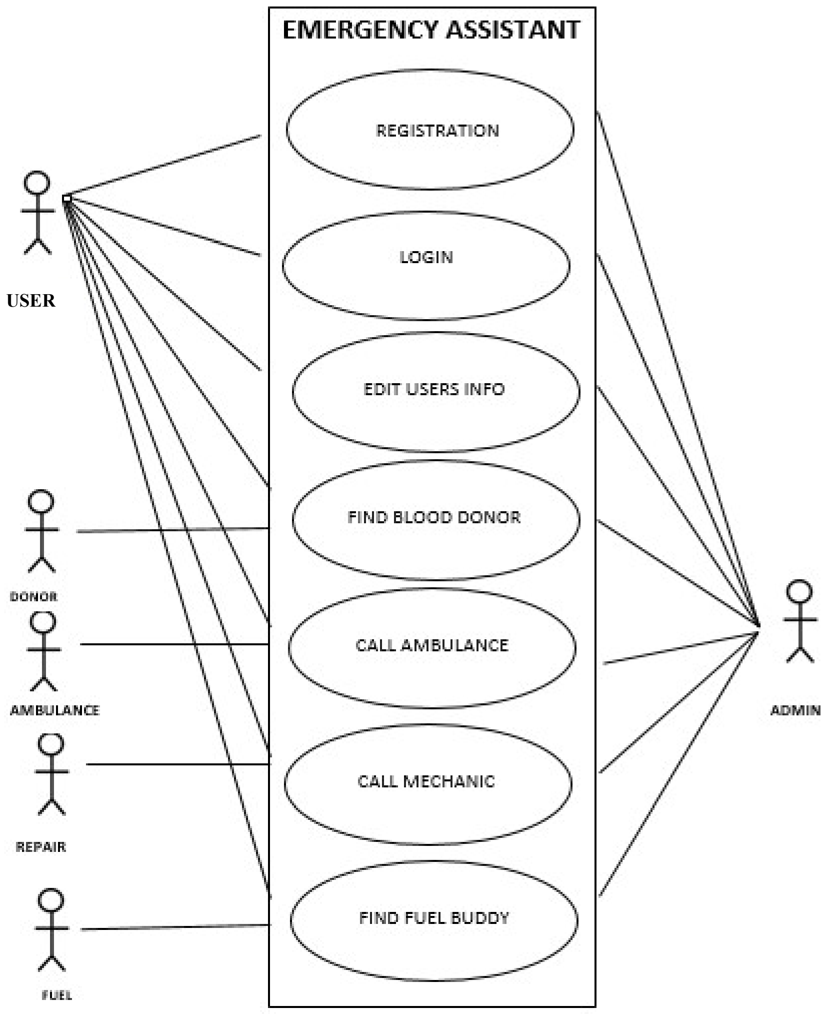
A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modelling Language), a standard notation forth modelling of real world objects and systems.

A use case diagram contains four components.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their roles.
- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points



4.2.2 CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

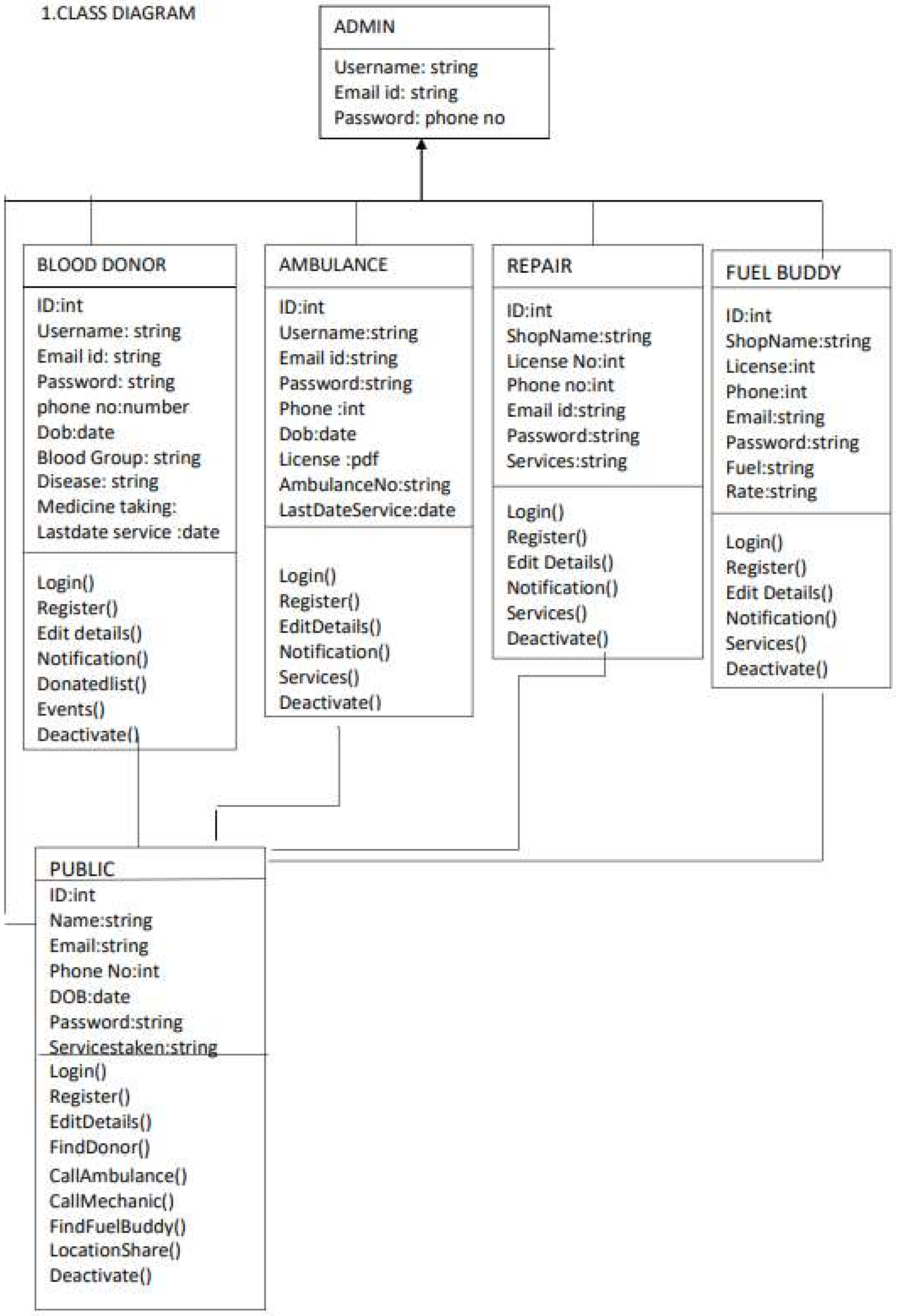
Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram

The following points should be remembered while drawing a class diagram –

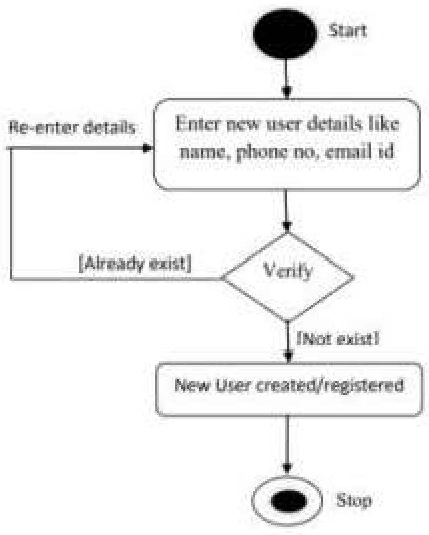
- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.
- Use notes whenever required to describe some aspect of the diagram. At the end of the drawing, it should be understandable to the developer/coder.

1.CLASS DIAGRAM



4.2.3 ACTIVITY DIAGRAM

Activity Diagrams are used to illustrate the flow of control in a system and refer to the steps involved in the execution of a use case. We model sequential and concurrent activities using activity diagrams. So, we basically depict workflows visually using an activity diagram. An activity diagram focuses on condition of flow and the sequence in which it happens. We describe or depict what causes a particular event using an activity diagram. An activity diagram portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed. We can depict both sequential processing and concurrent processing of activities using an activity diagram. They are used in business and process modelling where their primary use is to depict the dynamic aspects of a system.


Start
Re-enter details
Enter new user details like name, phone no, email id
[Already exist]
Verify
[Not exist]
New User created/registered
Stop

4.2.4 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

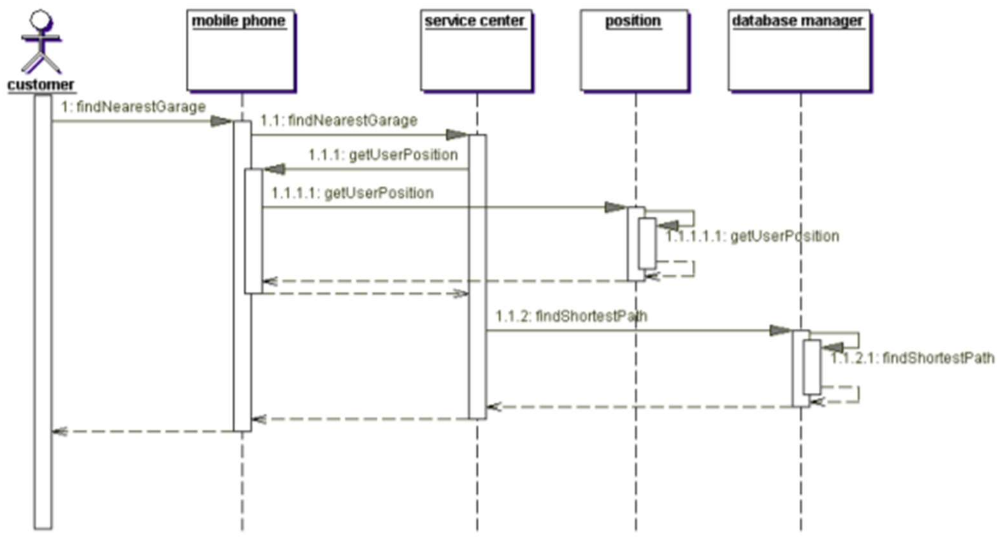
Sequence Diagram Notations –

i. **Actors** – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.

ii. **Lifelines** – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically, each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.

iii. **Messages** – Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

iv. **Guards** – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

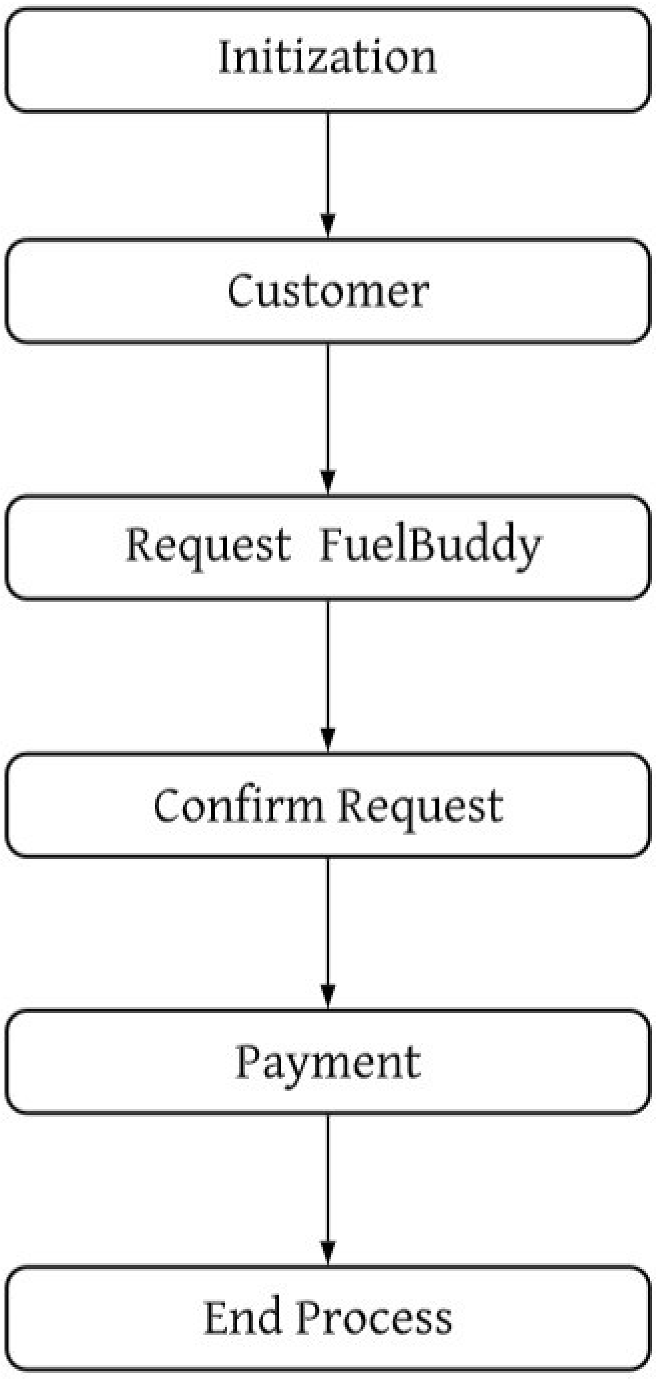

customer
mobile phone
service center
position
database manager
1: findNearestGarage
1.1: findNearestGarage
1.1.1: getUserPosition
1.1.1.1: getUserPosition
1.1.1.1.1: getUserPosition
1.1.2: findShortestPath
1.1.2.1: findShortestPath

4.2.5 COLLABORATION DIAGRAM

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other.

Notations of a Collaboration Diagram,

- **Objects**: The representation of an object is done by an object symbol with its name and class underlined, separated by a colon.In the collaboration diagram, objects are utilized in the following ways:
- The object is represented by specifying their name and class.It is not mandatory for every class to appear.A class may constitute more than one object.
- **Actors:** In the collaboration diagram, the actor plays the main role as it invokes the interaction. Each actor has its respective role and name. In this, one actor initiates the use case.
- **Links**: The link is an instance of association, which associates the objects and actors. It portrays a relationship between the objects through which the messages are sent. It is represented by a solid line. The link helps an object to connect with or navigate to another object,such that the message flows are attached to links.
- **Messages:** It is a communication between objects which carries information and includes a sequence number, so that the activity may take place. It is represented by a labeled arrow, which is placed near a link. The messages are sent from the sender to the receiver, and the direction must be navigable in that particular direction. The receiver must understand the message.

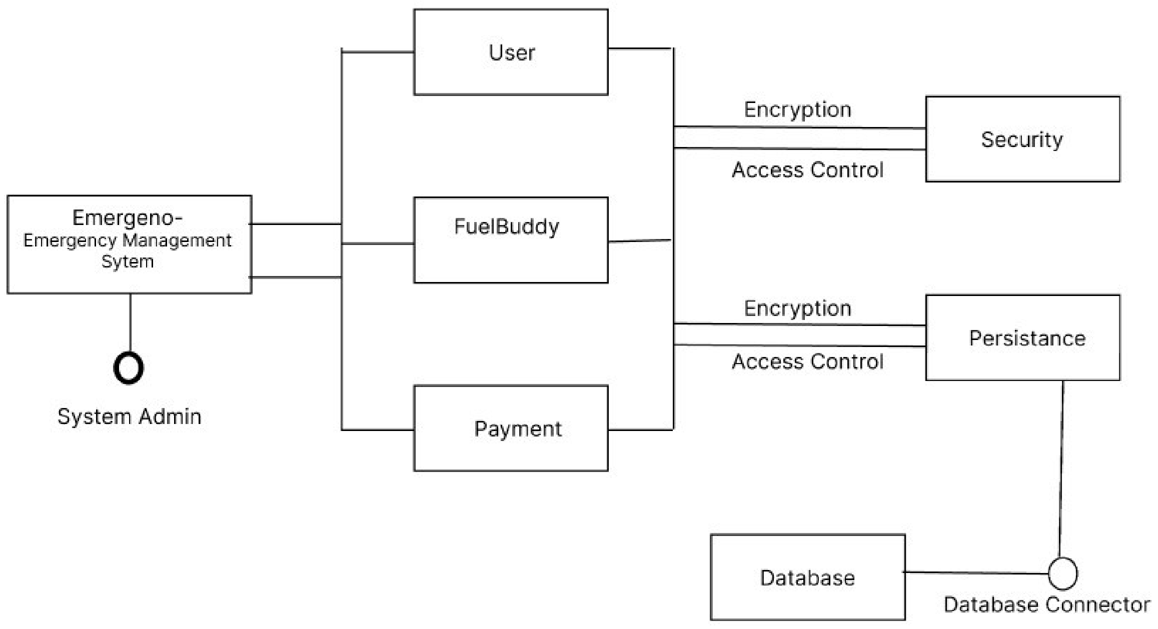

Initization
Customer
Request FuelBuddy
Confirm Request
Payment
End Process

4.2.6 COMPONENT DIAGRAM

Component diagram is a special kind of diagram in UML. The purpose is also different from all other diagrams discussed so far. It does not describe the functionality of the system but it describes the components used to make those functionalities. Component diagrams can also be described as a static implementation view of a system.

The purpose of the component diagram can be summarized as –

- Visualize the components of a system.
- Construct executables by using forward and reverse engineering.
- Describe the organization and relationships of the components.
-



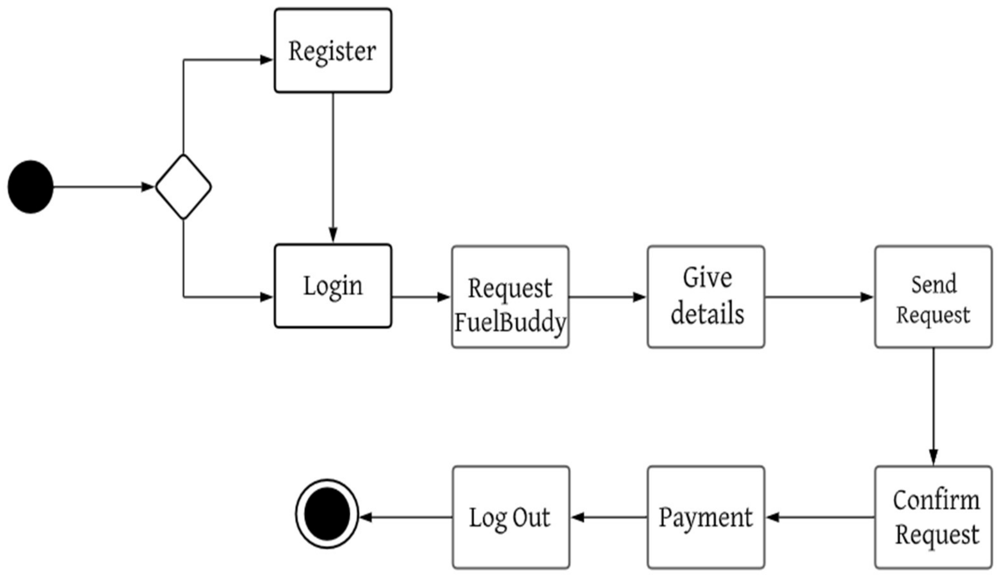
4.2.7 STATE CHART DIAGRAM

State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems.Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of Statechart diagram is to model lifetime of an object from creation to termination.

Following are the main purposes of using Statechart diagrams –

- To model the dynamic aspect of a system.
- To model the life time of a reactive system.
- To describe different states of an object during its life time.
- Define a state machine to model the states of an object.

Statechart diagram is used to describe the states of different objects in its life cycle. Emphasis is placed on the state changes upon some internal or external events.



4.2.8 DEPLOYMENT DIAGRAM

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.

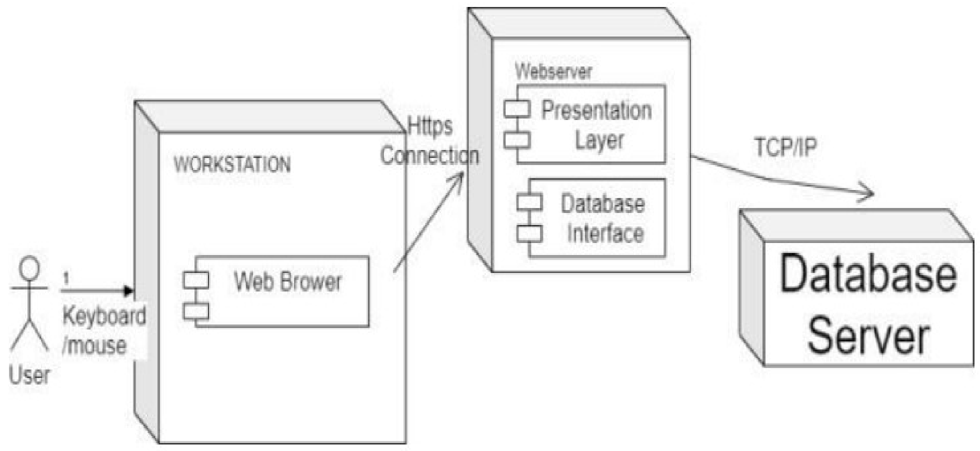
Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

The purpose of deployment diagrams can be described as –

• Visualize the hardware topology of a system.

• Describe the hardware components used to deploy software components.

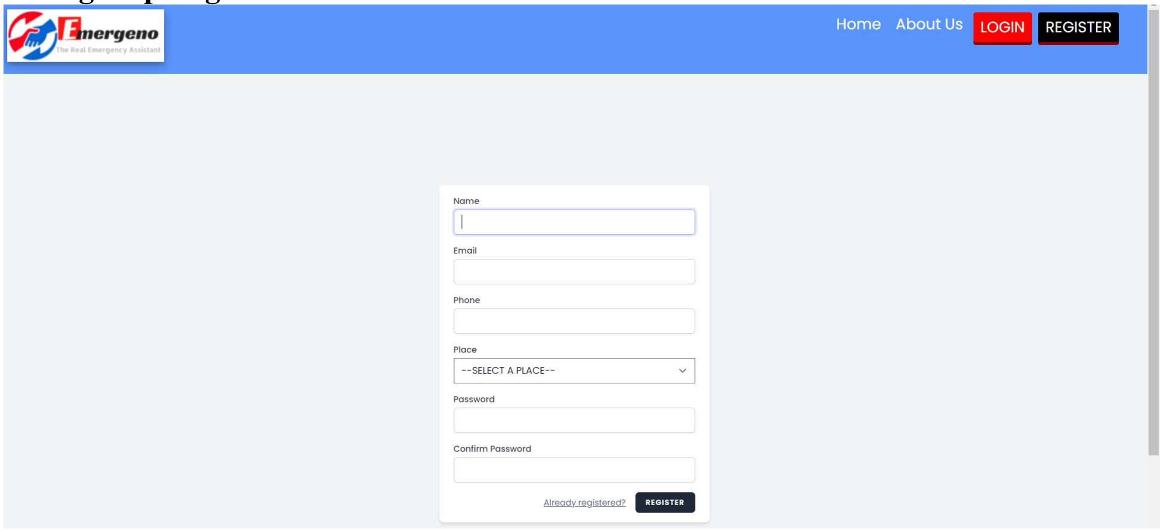
• Describe the runtime processing nodes



4.3 USER INTERFACE DESIGN

4.3.1 DESIGN PROTOTYPE

Sign Up Page



Name

Email

Phone

Place

--SELECT A PLACE--

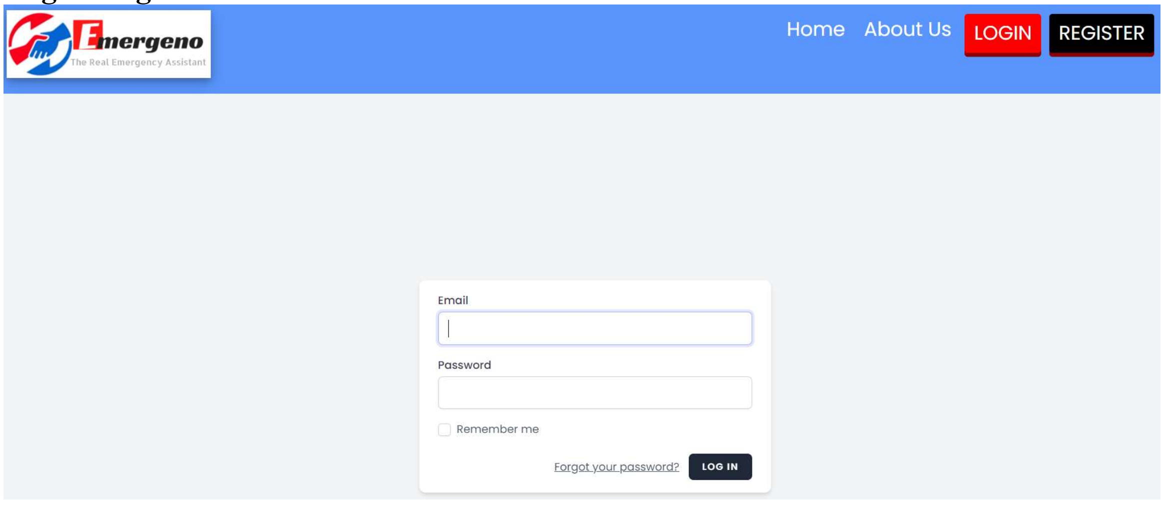
Password

Confirm Password

Already registered?

REGISTER

Log in Page



Email

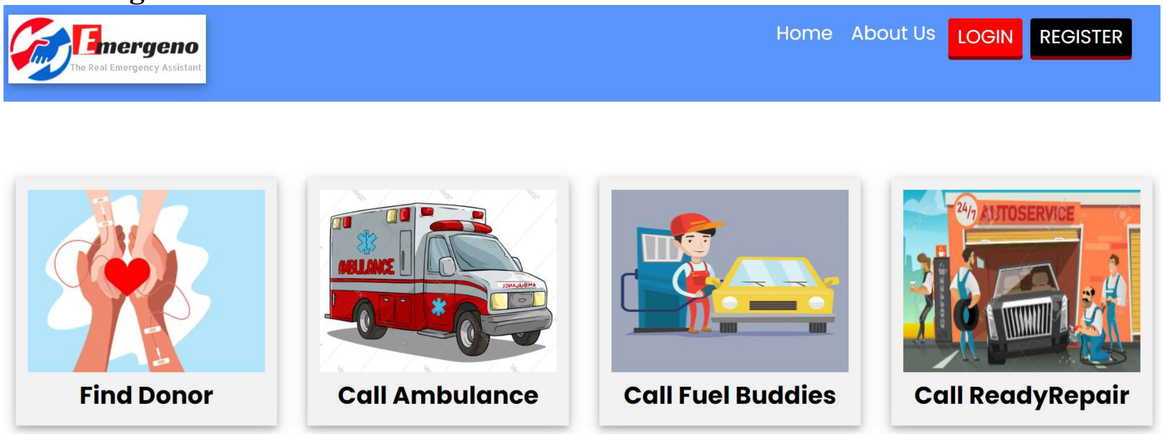
Password

Remember me

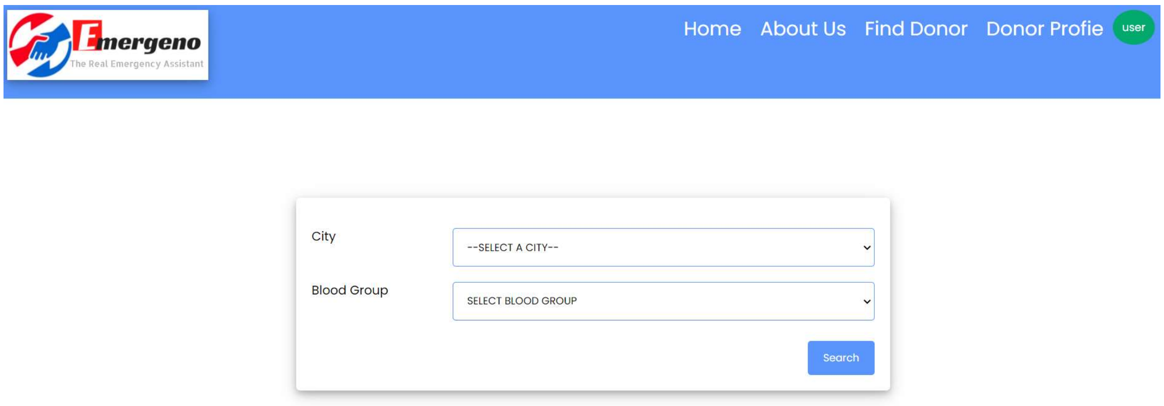
Forgot your password?

LOG IN

Home Page



Search Donor

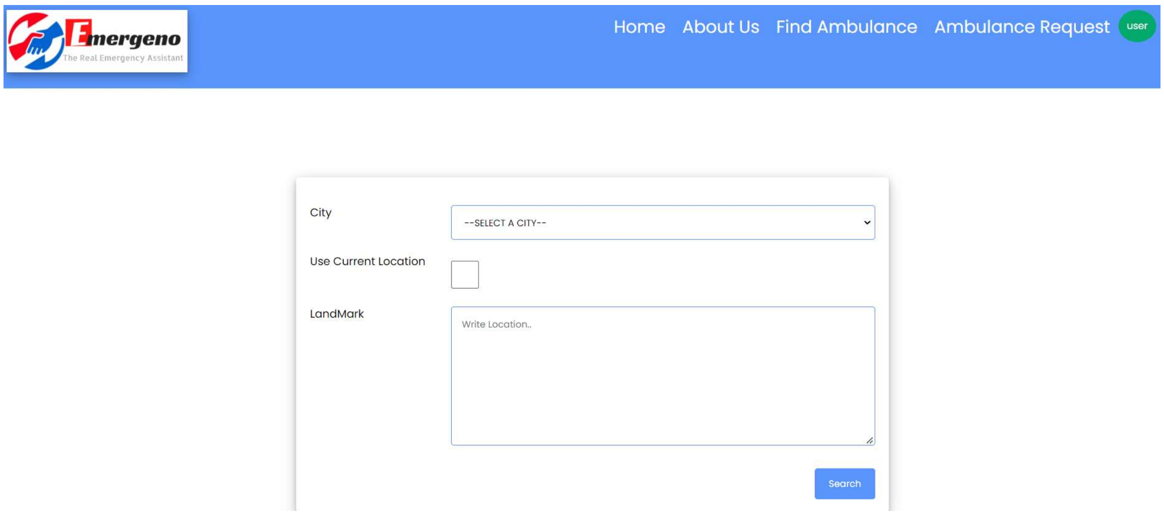


City --SELECT A CITY--

Blood Group SELECT BLOOD GROUP

Search

Search Ambulance



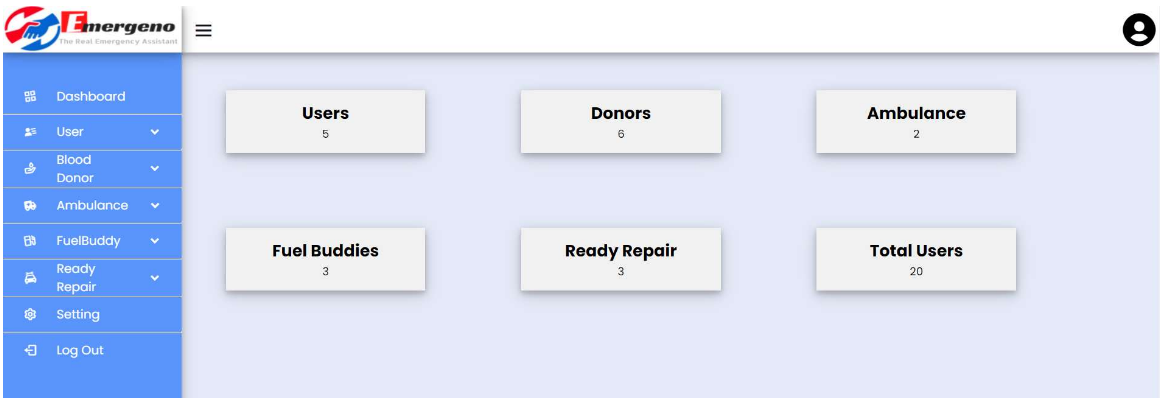
City --SELECT A CITY--

Use Current Location

LandMark Write Location..

Search

Admin DashBoard



Users 5	Donors 6	Ambulance 2
Fuel Buddies 3	Ready Repair 3	Total Users 20

4.4 DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is a two-level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS.

In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

• Data Integrity

• Data independence

Relational Database Management System (RDBMS)

A relational model represents the database as a collection of relations. Each relation resembles a table of values or file of records. In formal relational model terminology, a row is called a tuple, a column header is called an attribute and the table is called a relation. A relational database consists of a collection of tables, each of which is assigned a unique name. A row in a tale represents a set of related values.

Relations, Domains & Attributes

A table is a relation. The rows in a table are called tuples. A tuple is an ordered set of n elements. Columns are referred to as attributes. Relationships have been set between every table in the database. This ensures both Referential and Entity

Relationship Integrity. A domain D is a set of atomic values. A common method of specifying a domain is to specify a data type from which the data values forming the domain are drawn. It is also useful to specify a name for the domain to help in interpreting its values.

Relationships

• Table relationships are established using Key. The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
• Entity Integrity enforces that no Primary Key can have null values.
• Referential Integrity enforces that no Primary Key can have null values.
• Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key is Super Key and Candidate Keys.

Normalization

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table. All the tables have been normalized up to the third normal form. As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups at data and thereby avoids data redundancy which proves to be a great burden on the computer resources.

These include:

- Normalize the data.
- Choose proper names for the tables and columns.
- Choose the proper name for the data.

First Normal Form

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words, 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be donor by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

Second Normal Form

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attribute of the relation is fully dependent on its primary key alone.

Third Normal Form

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on another non-key attribute

TABLE DESIGN

Table No: 01

Table Name:tbl_users

Primary Key:Id

Table Description:To store user Login information

FIELD NAME	TYPE	Size	Description
Id	Int	10	Primary Key
Name	Varchar	50	Name of user
Email	Varchar	200	Email Id of user
Phone	Int	10	Phone No of user
Place	Varchar	200	Place of user
Password	Varchar	200	Password Of user
status	Int	10	Status of the user

Table No:-02

Table name:tbl_donor

Primary Key:Id

Table Description:To store donor details

FIELD NAME	TYPE	Size	CONSTRAINTS
Id	Int	10	Primary Key
uid	int	50	Foreign Key
dob	date	6	DOB of user
medlyf	Varchar	6	Taking medicine
weight	Int	5	Weight of user
gender	Varchar	8	Gender of user
bloodgrp	Varchar	20	Blood group
status	Int	5	Status of the user

Table_no :-03

Table_name:Tbl_ambu

Primary key:id

Table description:to store ambulance details

FIELD NAME	**TYPE**	**Size**	**CONSTRAINTS**
Id	Int	10	Primary Key
email	Varchar	200	Foreign Key
password	Varchar	200	Password
Vehicle_no	Varchar	15	Vehicle number
License_no	Varchar	15	License number
phone	Int	10	Phone number
place	Varchar	200	place
status	Int	4	Status of user

Table_no :-04

Table_name:Tbl_req_ambu

Primary key:id

Table description:to store ambulance details

FIELD NAME	**TYPE**	**Size**	**CONSTRAINTS**
id	Int	10	Primary Key
uid	Int	10	User id
aid	Int	10	Ambulance id
location	Varchar	200	location
longitude	Varchar	100	Longitude of location
latitude	Varchar	100	Latituse of location
crnt_loc	Int	4	Use current location
status	Int	4	Status of user

Table_no :-05

Table_name:Tbl_fuel

Primary key:id

Table description:to store fuelbuddy details

FIELD NAME	TYPE	Size	CONSTRAINTS
id	Int	10	Primary Key
name	Varchar	50	Name of company
email	Varchar	200	email
password	Varchar	200	Password
petrol_rs	Decimal	10,4	Petrol price
diesel_rs	Decimal	10,4	Diesel price
phone	Int	10	Phone number
place	Varchar	200	place
status	Int	4	Status of user

Table_no :-06

Table_name:Tbl_req_fuel

Primary key:id

Table description:to store fuel request

FIELD NAME	TYPE	Size	CONSTRAINTS
id	Int	10	Primary Key
uid	Int	10	User id
fid	Int	10	Fuel id
fuel	Varchar	50	Type of Fuel
price	Int	10	Price
location	Varchar	200	location
longitude	Varchar	100	Longitude of location
latitude	Varchar	100	Latituse of location
crnt_loc	Int	4	Use current location
status	Int	4	Status of user

Table_no :-07

Table_name:Tbl_repair

Primary key:id

Table description:to store gomechanic details

FIELD NAME	**TYPE**	**Size**	**CONSTRAINTS**
id	Int	10	Primary Key
name	Varchar	50	Name of company
email	Varchar	200	email
password	Varchar	200	Password
phone	Int	10	Phone number
place	Varchar	200	place
status	Int	4	Status of user

Table_no :-08

Table_name:Tbl_req_repair

Primary key:id

Table description:to store gomechanic request

FIELD NAME	**TYPE**	**Size**	**CONSTRAINTS**
id	Int	10	Primary Key
uid	Int	10	User id
rid	Int	10	Gomechanic id
service	Varchar	50	Type of service
price	Int	10	Price
location	Varchar	200	location
longitude	Varchar	100	Longitude of location
latitude	Varchar	100	Latituse of location
crnt_loc	Int	4	Use current location
status	Int	4	Status of user

Table_no :-09

Table_name:Tbl_repair_service

Primary key:id

Table description:to store gomechanic services

FIELD NAME	**TYPE**	**Size**	**CONSTRAINTS**
id	Int	10	Primary Key
rid	Int	10	Gomechanic id
service	Varchar	50	Type of service
status	int	10	Current status

Table_no :-10

Table_name:Tbl_payment

Primary key:id

Table description:to store gomechanic services

FIELD NAME	**TYPE**	**Size**	**CONSTRAINTS**
id	Int	10	Primary Key
Req_id	Int	10	Foreign key
amount	Varchar	50	Type of service
Payment_id	int	40	Payment id
api_id	Varchar	100	ID of used API
Payment done	int	10	Current status

CHAPTER 5
SYSTEM TESTING

5.1 INTRODUCTION

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the term's verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behaviour of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers-based system. Nothing is complete without testing, asits vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:
Testing is a process of executing a program with the intent of finding an error.

• A good test case is one that has high possibility of finding an undiscovered error.
• A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appears to be working according to the specification, that performance requirement appears to have been met.

5.2 TEST PLAN

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers are always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.
The levels of testing include:

- Unit testing
- Integration Testing
- Data validation Testing
- Output Testing

5.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm’s execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur.Boundary testing is the last task of unit testing step. Software often fails at its boundaries.
Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. After coding each module is tested and run individually. All unnecessary code were removed and ensured that all modules are working, and gives the expected result.

5.2.2 Integration Testing

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop. After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces.

5.2.3 Validation Testing or System Testing

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

5.2.4 Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

• Input Screen Designs
• Output Screen Designs

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

5.2.5 Selenium Testing

Selenium is one of the most widely used open-source Web UI (User Interface) automation testing suite. It was originally developed by Jason Huggins in 2004 as an internal tool at Thought Works. Selenium supports automation across different browsers, platforms and programming languages.

Selenium can be easily deployed on platforms such as Windows, Linux, Solaris and Macintosh. Moreover, it supports OS (Operating System) for mobile applications like iOS, windows mobile and android.

Selenium supports a variety of programming languages through the use of drivers specific to each language. Languages supported by Selenium include C#, Java, Perl, PHP, Python and Ruby. Currently, Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari. Selenium can be used to automate functional tests and can be integrated with automation test tools such as Maven, Jenkins, & Docker to achieve continuous testing. It can also be integrated with tools such as TestNG, & JUnit for managing test cases and generating reports.

Test Case 1

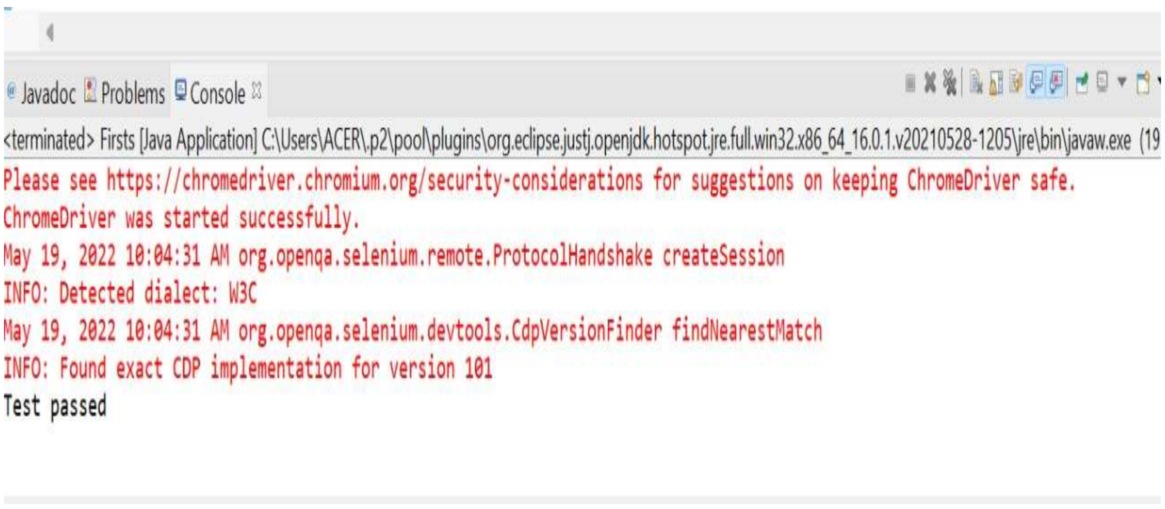
Project Name: Emergeno					
Login Test Case					
Test Case ID: Fun1			**TestDesigned By:**Alantina Mathew		
Test Priority: High			**Test Designed Date:**20-05-2022		
Module Name: Login			**TestExecutedBy**:Ms.Lisha Varghese		
Test Title: Verify login with valid email and password			**Test Execution Date:**23-05-2022		
Description: Test the Login Page					
Pre-Condition: User has valid email ID and Password					
Step	**Test Step**	**Test Data**	**Expected**	**Actual Result**	**Status**
1	Navigation Login Page		Page should be displayed	Page Displayed	Pass
2	Provide valid email	user@gmail.com	User should be able to Login	User Logged in and navigated to Dashboard	Pass
3	Provide valid password	12345678			
4	Click on button				
5	Provide Invalid Email or password	Email: us@gm.c Password: 123	User should not be able to Login	Message for enter valid emailID or password displayed	Pass
6	Provide Null credentials	Email: Null or Password: null			
7	Click on button				

Post-Condition: User is validated with database and successfully given the data in the application. The enquiry details by the user is added into the database.

Login page testcase

```
package testcase;
 import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import broswerimplement.DriverSetup;
public class Firsts {
public static WebDriver driver;
public static void main(String[] args) {
                    driver                                          =
DriverSetup.getWebDriver("http://localhost/final/login.blade.php");
            //login-Invalid case
          driver.findElement(By.name("email")).sendKeys("user@gmail.com");
            driver.findElement(By.name("password")).sendKeys("12345678");
            driver.findElement(By.name("submit")).click();

            String
actualUrl="http://localhost/library/Project/admin/dashboard/dashboard.php";
            String expectedUrl= driver.getCurrentUrl();
            if(actualUrl.equalsIgnoreCase(expectedUrl)) {
             System.out.println("Test passed"); } else { System.out.println("Test
failed"); }
```

OUTPUT



```
<terminated> Firsts [Java Application] C:\Users\ACER\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.1.v20210528-1205\jre\bin\javaw.exe (19
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
May 19, 2022 10:04:31 AM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
May 19, 2022 10:04:31 AM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found exact CDP implementation for version 101
Test passed
```

Test Case 2

Project Name: Emergeno	
User Registration Test Case	
Test Case ID: Fun_2	**Test Designed By:** Alantina Mathew
Test Priority(Low/Medium/High): Medium	**Test Designed Date:** 19-05-2022
Module Name: User entering contact details	**Test Executed By**:Ms.Lisha Varghese
Test Title : Verify enquiry details	**Test Execution Date:** 20-05-2022
Description: Test the enquiry details	
Pre-Condition: User	

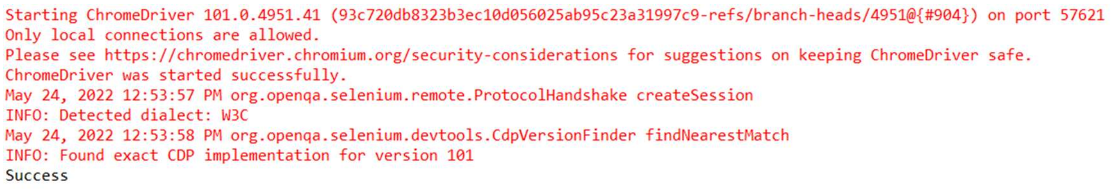
Step	**Test Step**	**Test Data**	**Expected Result**	**Actual Result**	**Status(Pass/Fail)**
1	Navigation to Enquiry details		Enquiry page should be displayed	Enquiry page displayed	Pass
2	Provide null information	Name: null	Validation message should be displayed	*Mandatory field message displayed	Pass
3	Provide Valid Details of user	All the mandatory enquiry details by the user	User should be able Give details correctly	Entered details successfully and home page displayed	Pass
4	Click on Submit button				

Post-Condition: User is validated with database and successfully given the data in the application. The enquiry details by the user is added into the database.

Contact form Test Case

```
package testcases;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;

import chromedriver.DriverSetup;

public class contactus {
public static WebDriver driver;

public static void main(String[] args) {
// TODO Auto-generated method stub

driver =  DriverSetup.getWebDriver("http://localhost/final/enquiry.blade.php");

//login-Invalid case
driver.findElement(By.name("fname")).sendKeys("Chikku Benny");
driver.findElement(By.name("email")).sendKeys("chikku@gmail.com");
driver.findElement(By.name("mobileno")).sendKeys("9908863080");
driver.findElement(By.name("subject")).sendKeys("Regarding to change phone no");
driver.findElement(By.name("description")).sendKeys("I want to know change my phone
no in your website?what should I do?.");
driver.findElement(By.name("submit1")).click();

if(driver.findElement(By.xpath("/html/body/div/div[1]/a/img")).isDisplayed()) {
System.out.println("Success");
}

driver.quit();
}
```

OUTPUT



```
Starting ChromeDriver 101.0.4951.41 (93c720db8323b3ec10d056025ab95c23a31997c9-refs/branch-heads/4951@{#904}) on port 57621
Only local connections are allowed.
Please see https://chromedriver.chromium.org/security-considerations for suggestions on keeping ChromeDriver safe.
ChromeDriver was started successfully.
May 24, 2022 12:53:57 PM org.openqa.selenium.remote.ProtocolHandshake createSession
INFO: Detected dialect: W3C
May 24, 2022 12:53:58 PM org.openqa.selenium.devtools.CdpVersionFinder findNearestMatch
INFO: Found exact CDP implementation for version 101
Success
```

CHAPTER 6
IMPLEMENTATION

6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover. The implementation state involves the following tasks:

- Careful planning.
- Investigation of system and constraints.
- Design of methods to achieve the changeover.

6.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to ensure that the resistance does not build up, as one has to make sure that:

• The active user must be aware of the benefits of using the new system.
• Their confidence in the software is built up.
• Proper guidance is imparted to the user so that he is comfortable in using the application

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won't take place.

6.2.1 User Training

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer-based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

6.2.2 Training on the Application Software

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the date entered. It should then cover information needed by the specific user/

group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy.

6.2.3 System Maintenance

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance. is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes

CHAPTER 7
CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The current system working technology is old fashioned and there is no usage of commonly used technologies like internet, digital money. The proposed system develop an integrated emergency service management system which is easily accessible to the public. This system will be a solution for all emergency needs like blood requirements, ambulance calls, fuel, vehicle repair, etc. . It is an automated software application with minimum cost. This system is more secure, reliable, and fast. This will help in better utilization of resources. User management tool in web-based project management system is a good appliance for keeping eye on the project and for giving rights to different users by system administrator in company. Finally, the whole system has been tested to ensure that everything functions correctly before the system processes actual data and produces information that people will rely on.

7.2 FUTURE SCOPE

- The proposed system is designed in such a way that the payment should be done in online mode.
- Customers can able to do advanced search options
- Data security can be enhanced.
- Root Optimization.
- Ensures Women Safety

CHAPTER 8

BIBLIOGRAPHY

REFERENCES

• Gary B. Shelly, Harry J. Rosenblatt, "System Analysis and Design", 2009.

• Roger S Pressman, "Software Engineering", 1994.

• Pankaj Jalote, "Software engineering: a precise approach", 2006.

• IEEE Std 1016 Recommended Practice for Software Design Descriptions.

WEBSITES

- www.w3schools.com
- www.jquery.com
- https://fuelbuddy.in/
- https://gomechanic.in/car-repair-near-me
- https://www.meddcoambulance.com/
- https://blooddonorskerala.org/

CHAPTER 9
APPENDIX

9.1 Sample Code

9.2.1 MainController.php

```
<?php

namespace App\Http\Controllers;
use App\Models\DonorReg;
use App\Models\User;

use Illuminate\Support\Facades\Hash;
use Illuminate\Http\Request;
use Carbon\Carbon;
use Stevebauman\Location\Facades\Location;
class MainController extends Controller
{


  public static function pswd(Request $request){

    $userID=auth()->user()->id;
    $find=  User::where('id',$userID)
    ->get();

    //dd($find);
  echo($find);


  }

    public function registerdonor(Request $request) {
        //dd($request);
        $request->validate([
          'dob' => ['required'],
          'med' => ['required'],
          'kg' => ['required','numeric'],
          'gender' =>['required'],
          'bloodgrp' => ['required'],
      ]);
      $birthDate=$request->input('dob');
      $age = \Carbon\Carbon::parse($birthDate)->age;
      $medi=$request->input('med');
      $weight=$request->input('kg');
      $interest=$request->input('grole');
      if($age>=18 && $medi=='No' && $weight>='40' && $interest=='Yes'){
        $id = auth()->user()->id;
        $donor = new DonorReg;
        $donor->uid = $id;
        $donor->dob = $request->input('dob');
        $donor->medlyf = $request->input('med');
        $donor->weight = $request->input('kg');
        $donor->gender = $request->input('gender');
        $donor->donor = $request->input('grole');
        $donor->bloodgrp = $request->input('bloodgrp');
        $donor->status = "1";
        $donor->save();
        echo '<script>alert("Mes")</script>';
        return redirect('/');

      }
      else{
        if($age<18){
          echo '<script>alert("Not Eligible!!age is below 18")</script>';
          return redirect()->back()->with('message','Not Eligible!!age is
below 18');
        }elseif($medi=='Yes'){
          echo '<script>alert("Not Eligible!!You are taking
medicine")</script>';
          return redirect()->back()->with('message','Not Eligible!!You are
taking medicine');
        }elseif($weight<'40'){
          echo '<script>alert("Not Eligible!!weight is below
40")</script>';
          return redirect()->back()->with('message','Not Eligible!!weight
is below 60');
        }elseif($interest=='No'){
          echo '<script>alert("Not Eligible!!coz you are not
interested")</script>';
          return redirect()->back()->with('message','Not Eligible!!coz you
are not interested');
        }else{
          echo '<script>alert("Error !!Try Again")</script>';
          return redirect()->back()->with('message','Error !!Try Again');
        }


      }

        }
        public function searchdonor(Request $request) {
           $userID=auth()->user()->id;
          $request->validate([
            'place' => ['required'],
            'bloodgrp' => ['required'],

        ]);
          $place=$request->input('place');
          $blood=$request->input('bloodgrp');
          $altblood='O Negative';
          //$userID=auth()->user()->id;
          $donor=DonorReg::join('users', 'users.id', '=', 'tbl_donor.uid')
          ->where([['users.place','=', $place],['users.id','<>',
$userID],['tbl_donor.status','=','1'],['tbl_donor.bloodgrp','=',$blood]])


          ->orWhere([['users.place','=', $place],['users.id','<>',
$userID],['tbl_donor.status','=','1'],['tbl_donor.bloodgrp','=',$altblood]]
)

          ->get(['tbl_donor.bloodgrp', 'users.place','users.phone']);
          $userID=auth()->user()->id;
          $find=  DonorReg::where('uid','=',$userID)->first();
          //dd($find);
          if($find===null){
            $status = 0;
            //alert("fyfy");
            return view('/donorlist',['a'=>$status,'donor'=>$donor]);
            die();
        }else{
          $status = 1;
         // alert("fyfy");
          return view('/donorlist',['a'=>$status,'donor'=>$donor]);
          die();
        }

         // return view('/donorlist',['donor'=>$donor]);
        }
        public static function check_reg_donor(Request $request){
          //$userID=auth()->user()->id;
          $userID=auth()->user()->id;
          $find=  DonorReg::where('uid','=',$userID)->first();
          //dd($find);
          if($find===null){
            $status = 0;
            //alert("fyfy");
            return view('/search_d',['a'=>$status]);
            die();
        }else{
          $status = 1;
         // alert("fyfy");
          return view('/search_d',['a'=>$status]);
          die();
        }
      }
        public static function donor2(Request $request){
```

```
      //$userID=auth()->user()->id;
      $userID=auth()->user()->id;
      $find=  DonorReg::where('uid','=',$userID)->first();
      //dd($find);
      if($find===null){
        $status = 0;
        //alert("fyfy");
        return view('/donor_profile',['a'=>$status]);
        die();
    }else{
      $sql=  DonorReg::where('uid','=',$userID)->get();
      $status = 1;
     // alert("fyfy");
      return view('/donor_profile',['a'=>$status,'alldata'=>$sql]);
      die();
    }

    }
    public static function donor3(Request $request){
      //$userID=auth()->user()->id;
      $userID=auth()->user()->id;
      $find=  DonorReg::where('uid','=',$userID)->first();
      //dd($find);
      if($find===null){
        $status = 0;
        //alert("fyfy");
        return view('/edit_donor',['a'=>$status]);
        die();
    }else{
      //$sql=  DonorReg::where('uid','=',$userID)->get();
      $status = 1;
     // alert("fyfy");
      return view('/edit_donor',['a'=>$status]);
      die();
    }

    }
    public static function donor4(Request $request){
      //$userID=auth()->user()->id;
      $userID=auth()->user()->id;
      if($request->input('med')=="Yes" || $request->input('dc')=="No")
      {


      $update= DonorReg::where('uid','=',$userID)->first();
      $update->weight=$request->input('weight');
      $update->medlyf=$request->input('med');
      $update->donor=$request->input('dc');
```

```
          $update->status=0;
          $update->save();
          }else{

            $update= DonorReg::where('uid','=',$userID)->first();
            $update->weight=$request->input('weight');
            $update->medlyf=$request->input('med');
            $update->donor=$request->input('dc');
            $update->status=1;
            $update->save();

          }

return redirect()->back()->with('message','Successfully Updated');

        }

        public static function check_reg_donor1(Request $request){
          $userID=auth()->user()->id;
          $find=  DonorReg::where('uid','=',$userID)->first();
          //dd($find);
          if($find===null){
            $status = 0;
            //alert("fyfy");
            return view('/donor_reg',['a'=>$status]);
            die();
        }else{
          $status = 1;
         // alert("fyfy");
          return view('/donor_reg',['a'=>$status]);
          die();
        }

        }
        public static function check_reg_donor2(Request $request){
          $userID=auth()->user()->id;
          //$alldata=User::all();
          $find=  DonorReg::where('uid','=',$userID)->first();
          //dd($find);
          if($find===null){
            $status = 0;
            //alert("fyfy");
            return view('/view_donor',['allarray'=>$find,'a'=>$status]);
            die();
        }else{
          $status = 1;
         // alert("fyfy");
          return view('/view_donor',['allarray'=>$find,'a'=>$status]);
          die();
        }

        }
}
```

9.2.2 HomeController.php

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        return view('home');
    }
}
```

9.2.3FuelController.php

```
<?php

namespace App\Http\Controllers;
use App\Models\Fuel;
use App\Models\User;
use App\Models\Req_fuel;
use App\Models\Fuel_loc;
use Carbon\Carbon;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Validation\Rules;
```

```
class FuelController extends Controller
{
    public function reg(Request $request){
        $em=$request->input('email');
        //$string=implode(",",$request->input('place'));


       $log=Fuel::where('email',$em)

       ->count();
        if($log>0)
        {
            return redirect()->back()->with('message','user already
exist');
        }else{
        $psw=$request->input('password');
        $cpsw=$request->input('password_confirmation');
        if($psw==$cpsw){
        $adduser=new Fuel();
        $adduser->name=$request->input('name');
        $adduser->email=$request->input('email');
        $adduser->phone=$request->input('phone');
        $adduser->password=$request->input('password');
        $adduser->save();

        $id=$adduser->id;
        foreach($request->input('place') as $a){
             $addloc=new Fuel_loc();
             $addloc->fid=$id;
             $addloc->place=$a;
             $addloc->save();
        }
        return view('/fuel_login');
        }else{
            return redirect()->back()->with('message','Password should be
same');
        }
        }
        }
    public function log(Request $request){
        $em=$request->input('email');
        $ps=$request->input('password');
       $login=Fuel::where('email',$em)
       ->where('password',$ps)
       ->count();
       $logf=Fuel::where('email',$em)
       ->where('password',$ps)
       ->get();
```

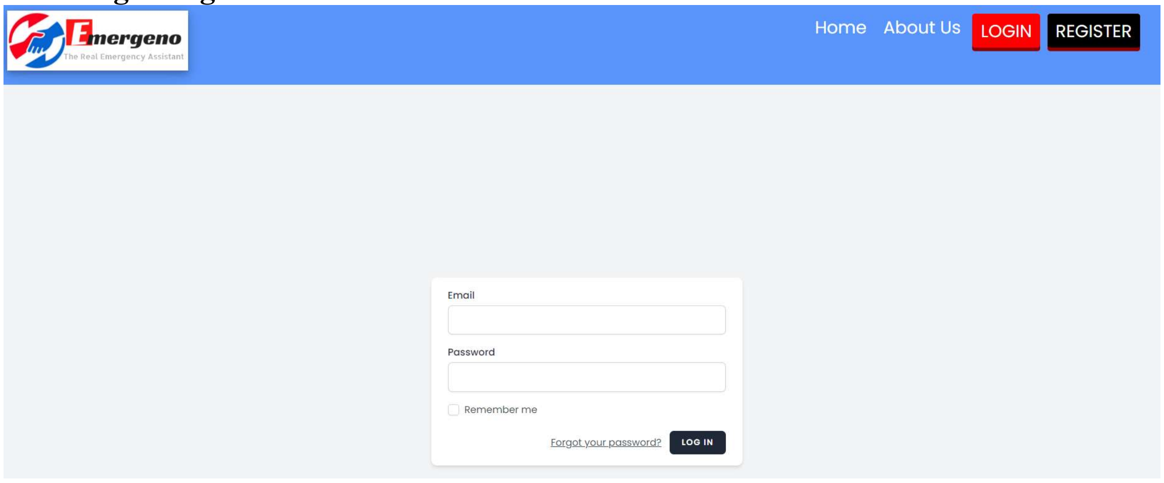
```
        if($login>0)
        {


               // Session::put('name' , $name);
                session()->put('fuel_name',$em);
                session()->save();
                foreach($logf as $a)
                {
                    $lf= $a->id;
                    $petrol=$a->petrol_rs;
                    $disel=$a->disel_rs;
                }
                session()->put('fuel_id',$lf);
                session()->save();
                //$request->session()->put('loginid', $login[0]->email);
                return $this->dash();

        }
        else
        {
            session()->flash('statusn',"Deleted");

            return redirect()->back()->with('status','Username or Password
is wrong');
        }
    }
    public function dash(){
            $lf=session('fuel_id');
            $logf=Fuel::where('id',$lf)

            ->get();
            if($logf)
            {
                    foreach($logf as $a)
                    {

                        $petrol=$a->petrol_rs;
                        $disel=$a->disel_rs;
                    }

                    //$request->session()->put('loginid', $login[0]-
>email);

                    }

                    $find=Req_fuel::where('fid','=',$lf)-
>where('created_at', '>=', Carbon::now()->subDay())->get();
```

```
                                //dd($find);
                if(count($find)>0){
                    $sql=Req_fuel::join('tbl_fuel','tbl_fuel.id','=','tbl_r
eq_fuel.fid')
                    ->join('users','users.id','=','tbl_req_fuel.uid')
                    ->where('tbl_req_fuel.created_at', '>=', Carbon::now()-
>subDay())
                    ->where('tbl_req_fuel.fid','=',$lf)
                    -
>get(['tbl_req_fuel.location','tbl_req_fuel.status','tbl_req_fuel.crnt_loc'
,'tbl_req_fuel.id','tbl_req_fuel.place','tbl_req_fuel.fuel','users.phone'])
;
                    //dd($sql);
                    if($sql){

                        return
view('/dash_fuel',['petrol'=>$petrol,'disel'=>$disel,'a'=>$sql]);
                }else{
                    $b="failed";
                    $find=[];
                   return
view('/dash_fuel',['petrol'=>$petrol,'disel'=>$disel,'a'=>$find,'b'=>$b]);
                }

                }else{
                    $b="utter failed";
                    $find=[];

                    return
view('/dash_fuel',['petrol'=>$petrol,'disel'=>$disel,'a'=>$find,'b'=>$b]);
                }

                }
public function view_loc($id){
    $find=Req_Fuel::where('id',$id)->first()
    ->get(['latitude','longitude']);
    return view('/app_fuel',['a'=>$find]);
}
    public function petrol(Request $request){
        echo "abhbkjbkjb";
        $lf=session('fuel_id');

        //echo($aa);
        $price=$request->input('petrol');
        $update=Fuel::where('id',$lf)->first();
        $update->petrol_rs=$price;
        $update->save();
        return $this->dash();
        }
    public function disel(Request $request){
        echo "abhbkjbkjb";
        $lf=session('fuel_id');
        $price=$request->input('disel');

        $update=Fuel::where('id','=',$lf)->first();
            $update->disel_rs=$price;
            $update->save();
        return $this->dash();
        }



    public function fuelBuddy(Request $request){
        $userID=auth()->user()->id;

        $fuel=$request->input('fuel');

        $place=$request->input('place');

        session(['fuel_loc' => $request->input('location')]);
        session(['fuel_place' => $request->input('place')]);
        session(['fuel_type' => $request->input('fuel')]);
        session(['latitude' => $request->input('lat')]);
        session(['longitude' => $request->input('lng')]);
        session(['crnt' => $request->input('chk')]);

        // $sql=Fuel::where('place', 'like',$place)
        //     ->where('petrol_rs', '<>',NULL)
        //     ->get(['id']);
        //     dd($sql);

            $find=Req_fuel::where('uid',$userID)
            ->where('tbl_req_fuel.created_at', '>=', Carbon::now()-
>subDay())->count();
            if($find>0){
                   if($fuel=='Petrol'){
                    $sql=Fuel::join('tbl_req_fuel','tbl_req_fuel.fid','<>',
'tbl_fuel.id')
                    ->join('fuel_loc','fuel_loc.fid', '=','tbl_fuel.id')
                    ->where('fuel_loc.place', '=',$place)
                   ->where('tbl_fuel.status','=','1')
                   ->where('tbl_fuel.petrol_rs','<>',0.00)
                    ->where('tbl_req_fuel.created_at', '>=', Carbon::now()-
>subDay())
```

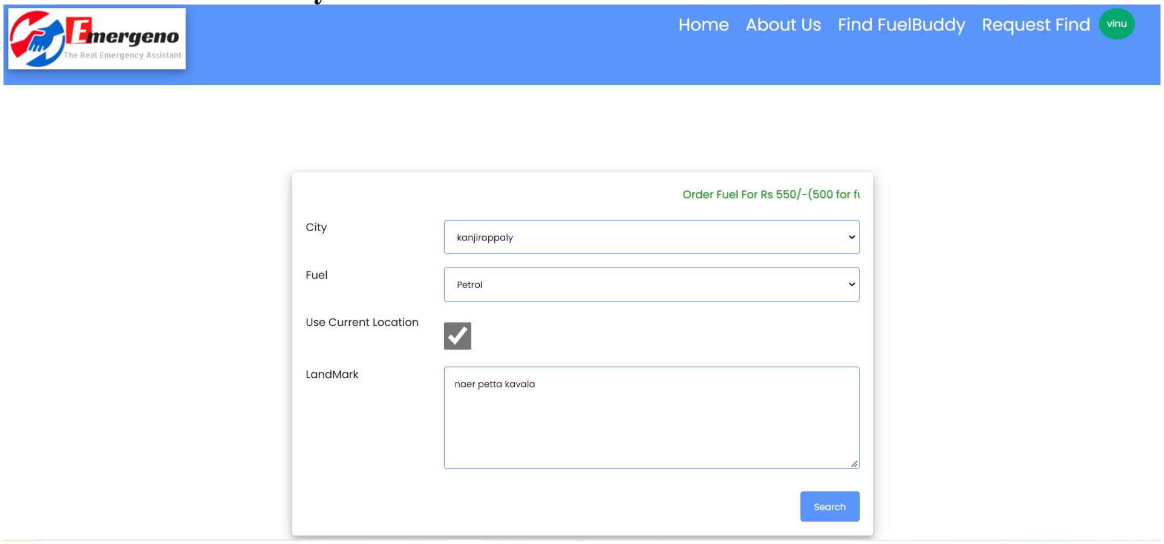
```
                    ->distinct(['tbl_fuel.id'])-
>get(['tbl_fuel.id','fuel_loc.place','tbl_fuel.petrol_rs','tbl_fuel.phone',
'tbl_fuel.name']);
                    //dd($sql);
                     return view('/fuellist',['a'=>$sql,'fuel'=>$fuel]);
                   }else{
                    $fuel='Diesel';
                    $sql=Fuel::join('tbl_req_fuel','tbl_req_fuel.fid','<>',
'tbl_fuel.id')
                    ->join('fuel_loc','fuel_loc.fid', '=','tbl_fuel.id')
                    ->where('fuel_loc.place', '=',$place)
                   ->where('tbl_fuel.status','=','1')
                   ->where('tbl_fuel.disel_rs','<>',0.00)
                    ->where('tbl_req_fuel.created_at', '>=', Carbon::now()-
>subDay())

                    ->distinct(['tbl_fuel.id'])-
>get(['tbl_fuel.id','fuel_loc.place','tbl_fuel.disel_rs','tbl_fuel.phone','
tbl_fuel.name']);
                    return view('/fuellist',['a'=>$sql,'fuel'=>$fuel]);
                   }
            }else{
                if($fuel=='Petrol'){
                $sql=Fuel::join('fuel_loc','fuel_loc.fid',
'=','tbl_fuel.id')
                ->where('fuel_loc.place', '=',$place)
                        ->where('tbl_fuel.petrol_rs', '<>',0.00)
                        ->where('tbl_fuel.status','=','1')
                        ->distinct(['tbl_fuel.id'])-
>get(['tbl_fuel.id','fuel_loc.place','tbl_fuel.petrol_rs','tbl_fuel.phone',
'tbl_fuel.name']);
                        return view('/fuellist',['a'=>$sql,'fuel'=>$fuel]);
                    }else{
                    $sql=Fuel::join('fuel_loc','fuel_loc.fid',
'=','tbl_fuel.id')
                   ->where('fuel_loc.place', '=',$place)
                        ->where('tbl_fuel.disel_rs', '<>',0.00)
                        ->where('tbl_fuel.status','=','1')
                        ->distinct(['tbl_fuel.id'])-
>get(['tbl_fuel.id','fuel_loc.place','tbl_fuel.disel_rs','tbl_fuel.phone','
tbl_fuel.name']);
                        return view('/fuellist',['a'=>$sql,'fuel'=>$fuel]);
                    }
             }
          }

    }
```

9.2 ScreenShot

9.2.1 Login Page



9.2.2 Search FuelBuddy



Order Fuel For Rs 550/-(500 for f

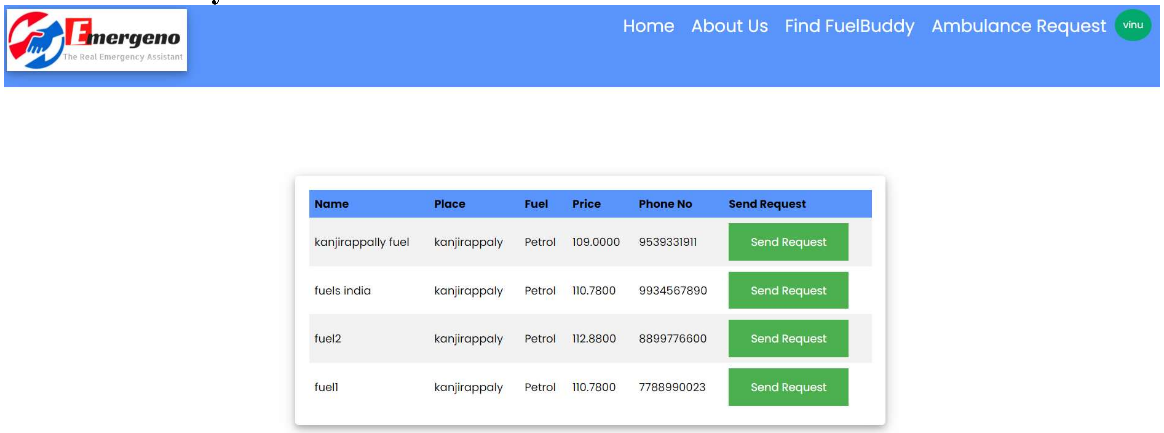
City: kanjirappaly

Fuel: Petrol

Use Current Location: ✓

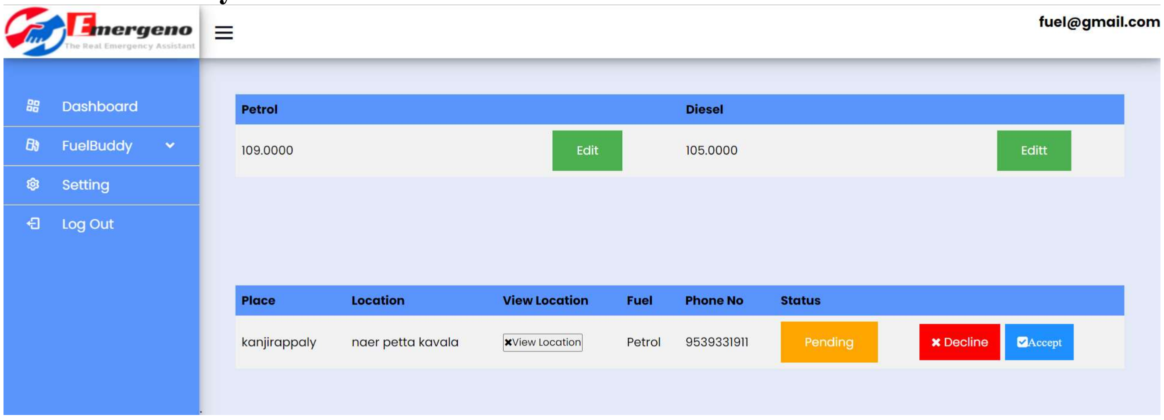
LandMark: naer petta kavala

9.2.3 FuelBuddy List



Name	Place	Fuel	Price	Phone No	Send Request
kanjirappally fuel	kanjirappaly	Petrol	109.0000	9539331911	Send Request
fuels india	kanjirappaly	Petrol	110.7800	9934567890	Send Request
fuel2	kanjirappaly	Petrol	112.8800	8899776600	Send Request
fuel1	kanjirappaly	Petrol	110.7800	7788990023	Send Request

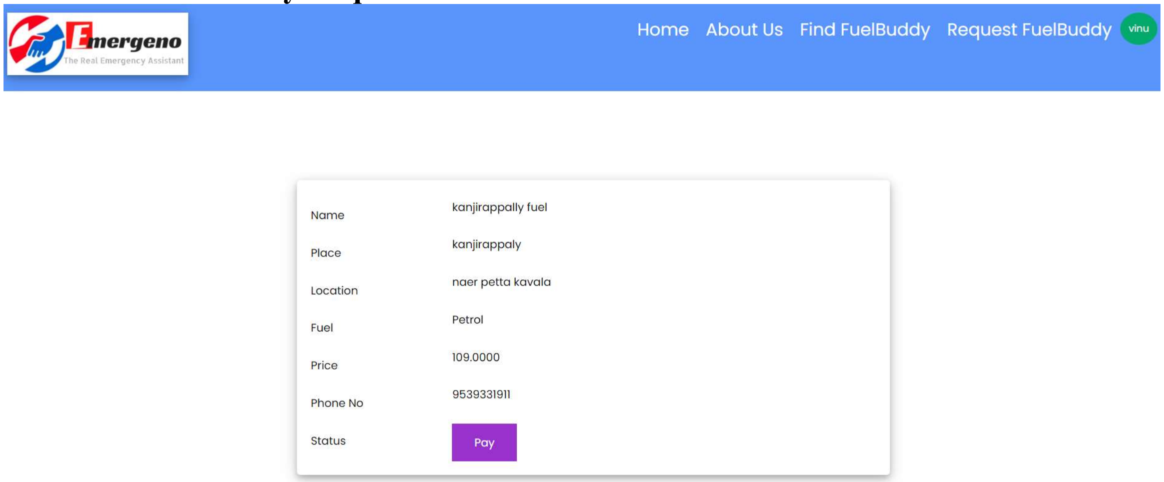
9.2.4 Fuel Buddy Dashboard



Petrol		Diesel	
109.0000	Edit	105.0000	Editt

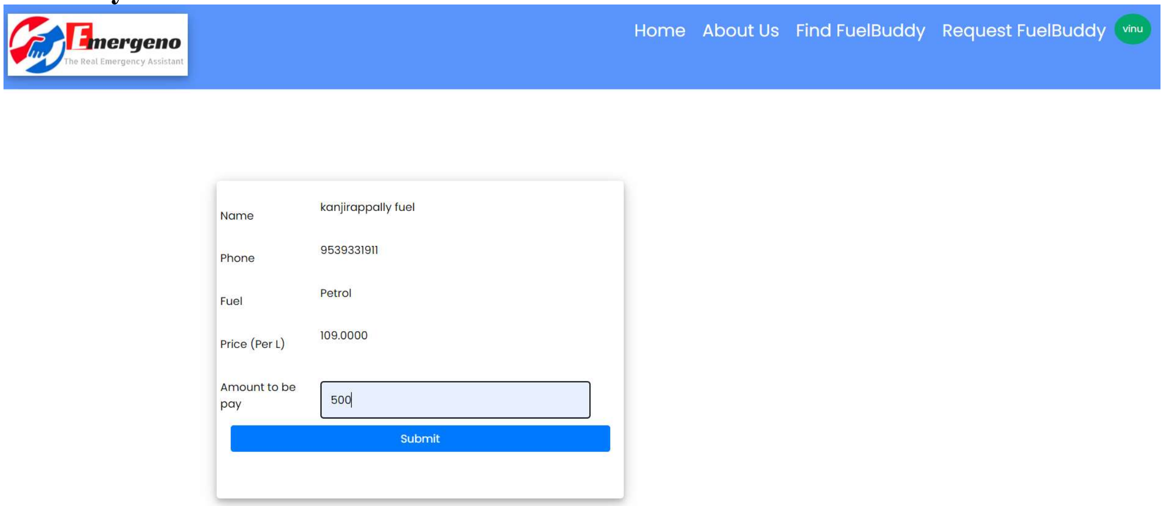
Place	Location	View Location	Fuel	Phone No	Status		
kanjirappaly	naer petta kavala	View Location	Petrol	9539331911	Pending	Decline	Accept

9.2.5 User FuelBuddy Request Status



Name	kanjirappally fuel
Place	kanjirappaly
Location	naer petta kavala
Fuel	Petrol
Price	109.0000
Phone No	9539331911
Status	Pay

9.2.6 Payment



Name	kanjirappally fuel
Phone	9539331911
Fuel	Petrol
Price (Per L)	109.0000
Amount to be pay	500