# 

# VEHICLE BREAKDOWN ASSISTANCE

Project submitted to the Bharathiar University in partial fulfillment of the

Requirements for the Award of the Degree of

## BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

Submitted by

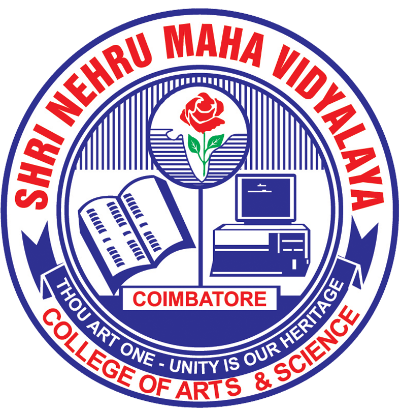
**PRAVEENKUMAR.N**

**(Reg.no: 2026J0684)**

Under the guidance of

## Dr. M. JAYA KUMAR, M.CA.,M.Phil.,Ph.D.,

## (Assistant Professor)

****

# DEPARTMENT OF INFORMATION TECHNOLOGY

# (SNMV)

# SHRI NEHRU MAHA VIDYALAYA

# College of arts and science

Approved by AICTE New Delhi &Affliated to Bharathiar University

An ISO 9001: 2008 Certified Institution

Malumachampatti,

Coimbatore-641050.

**APRIL-2023**

**ACKNOWLEDGEMENT**

I am extremely grateful to our principal **Dr. B. SUBRAMANI, M.Sc., PGDCA., B.Ed., MCA., M.Phil., MBA., Ph.D.,** SNMV college of Arts and Science, Coimbatore for his indirect help and guidance.

I wish to thank **Mr. R. NANDHA KUMAR M.Sc., M.Phil.,** Head, Department of Information Technology, SNMV College of Arts and Science, Coimbatore for his wonderful suggestion and guidance.

I wish to thank my guide **Dr. M. JAYA KUMAR.,** **M.CA., M.Phil., Ph.D.,** Assistant professor in Department of Information Technology, SNMV College of Arts and Science, Coimbatore for his wonderful suggestion and guidance.

I am very glad to thank my **parents** and my **friends** for their moral support and timely help.

**CERTIFICATE**

This is to certify that the project entitled

# “VEHICLE BREAKDOWN ASSISTANCE”

# is a record of bonafide work done by

# PRAVEENKUMAR.N

# (Reg.No:2026J0684)

# Submitted to Bharathiar University in partial fulfillment

# Of the requirements for the award of the degree of

## BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

SIGNATURE OF THE GUIDE SIGNATURE OF THE HOD

**(Dr. M. JAYA KUMAR) (Mr. R. NANDHA KUMAR)**

**PRINCIPAL**

**(Dr. B. SUBRAMANI)**

**S**ubmitted for the Viva-Voice Examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**DECLARATION**

I hereby declare that the project work entitled **“VEHICLE BREAKDOWN ASSISTANCE”** submitted to Bharathiar University Coimbatore, in partial fulfillment of the requirements for the degree of **B.Sc INFORMATION TECHNOLOGY** is the record of original work done by me under the supervision of **Dr. M. JAYA KUMAR.,** **M.CA.,M.Phil.,Ph.D.,** Assistant professor in Information Technology and it has not been previously formed the basis for the award of any Degree ,Diploma ,Associate ship ,Fellowship or other similar title to any other University, or body during the period of my study.

Place:

Date:

**SIGNATURE OF THE CANDIDATE**

**PRAVEENKUMAR.N**

**(Reg.No:2026J0684)**

**ABSTRACT**

The proposed web application helps to find mechanics easily and quickly. It is difficult to find mechanics nearby area wherever you are travelling. This system helps to overcome this issue by providing mechanic details in one click. Here the locator allows you to search mechanics from different locations. Admin is allowed to access and manage mechanic details. This online mechanic locator reduces work and can easily find the mechanics from various location. Reduces your time and cost.

The main objective is to provide a better service and to make the process easily and finally appointing a mechanic quickly. The System includes following modules are,

* Admin
* Mechanic
* Filters
* Chatbot
* Reports

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TITLE** | **PAGE NO** |
|  | **ACKNOWLEDGEMENT** |  |
|  | **ABSTRACT** |  |
| 1 | **INTRODUCTION** | 1 |
|  | 1.1 ABOUT THE PROJECT | 1 |
|  | 1.2 ORGANIZATION PROFILE | 1 |
|  | 1.3 SYSTEM SPECIFICATION | 2 |
|  | 1.3.1 HARDWARE SPECIFICATION | 2 |
|  | 1.3.2 SOFTWARE SPECIFICATION | 2 |
|  | 1.4 SOFTWARE FEATURES | 3 |
| 2 | **SYSTEM STUDY** | 7 |
|  | 2.1 EXISTING SYSTEM | 7 |
|  | 2.2 PROPOSED SYSTEM | 7 |
| 3 | **SYSTEM DESIGN** | 8 |
|  | 3.1 INPUT DESIGN | 8 |
|  | 3.2 OUTPUT DESIGN | 8 |
|  | 3.3 DATABASE DESIGN | 9 |

|  |  |  |
| --- | --- | --- |
| 4 | **SYSTEM DEVELOPMENT** | 10 |
|  | 4.1 DESCRIPTION OF MODULES | 10 |
| 5 | **SYSTEM TESTING** | 11 |
|  | 5.1 TESTING | 11 |
|  | 5.2 SYSTEM SECURITY | 12 |
| 6 | **SYSTEM IMPLEMENTATION** | 13 |
| 7 | **CONCLUSION** | 14 |
| 8 | **FUTURE ENHANCEMENT** | 15 |
| 9 | **BIBLIOGRAPHY** | 16 |
| 10 | **APPENDICES** | 17 |
|  | 10.1 DATAFLOW DIAGRAM | 17 |
|  | 10.2 ENTITY RELATIONSHIP DIAGRAM | 18 |
|  | 10.3 TABLE DESIGN | 19 |
|  | 10.4 SAMPLE SCREENS | 21 |
|  | 10.5 SAMPLE CODING | 35 |

**1. INTRODUCTION**

**1.1 ABOUT THE PROJECT**

A lot of people are facing difficulties getting help when their car breaks down on the road. Many of them do not have any car repair service providers’ contact number and could not get help as the car repair service providers might be far away from their locations. These problems are the motivations for the development of this project to help those who are in need when their car breaks down along the roads. This project will start with the analysis of the car breakdown incidents on the road.

Roadside emergencies are unpredictable and can be fatal, risky and stressful. This project is designed in that way to provide a solution for this situation. As part of the expected results, the proposed system connects car service providers and the Public through this system. If the car owner's transportation breaks down on any highway or federal road in any part, the owner could enter information with regards to the place of breakdown in the system using mobile phone, tablets. The system will automatically search for any mechanic or workshop nearest to the reported incident spot. The users are able to contact the mechanic to service the vehicle.

**1.2 ORGANIZATION PROFILE**

Jooba is a progressive software and website development concern with good record of developing various websites and application software. Currently the concern is engaged with Software developers, designers and marketing employees. The concern was started in 2008 by Mr. Chandramouli.

The upcoming technology solution provides customer service to achieve decisive advantages with end users and for both industrial and commercial customers as well as private end consumers. Jooba has customers all over from various places in Tamilnadu. Their customers include major textile manufacturers, export companies, industrial machinery, retailers. The concern is located in Coimbatore.

Jooba has great resources in terms of people, expertise and technology partnership and they have got a team of well experienced and motivated graphic designers, software engineers, system professionals and marketing executives. The web id is joobaa.com.

**1.3 SYSTEM SPECIFICATION**

**1.3.1 HARDWARE SPECIFICATION**

Processor : Intel Pentium i3.

RAM : 4 GB

Hard drive : 500 GB

Monitor : 17” Flat L.G color SVGA

Keyboard : Multimedia keyboard

Mouse : Logitech Optical scroll mouse

Print : Cannon LaserJet

**1.3.2 SOFTWARE SPECIFICATION**

Front-End Tool : Python

Back-End Tool : My Sql

Framework : Flask

**1.4 SOFTWARE FEATURES**

**Python**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

**Python Features**

* Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* Easy-to-read − Python code is more clearly defined and visible to the eyes.
* Easy-to-maintain − Python's source code is fairly easy-to-maintain.
* A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* Extendable − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* Databases − Python provides interfaces to all major commercial databases.
* GUI Programming − Python supports GUI applications that can be created and ported to many system

calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

* Scalable − Python provides a better structure and support for large programs than shell scripting.

**Flask**

Flask is a web application framework written in Python. It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Poocco. Flask is based on the Werkzeg WSGI toolkit and the Jinja2 template engine.Both are Pocco projects.

**WSGI**

The Web Server Gateway Interface (Web Server Gateway Interface, WSGI) has been used as a standard for Python web application development. WSGI is the specification of a common interface between web servers and web applications.

**Werkzeug**

Werkzeug is a WSGI toolkit that implements requests, response objects, and utility functions. This enables a web frame to be built on it. The Flask framework uses Werkzeg as one of its bases.

**jinja2**

jinja2 is a popular template engine for Python. A web template system combines a template with a specific data source to render a dynamic web page.

**BACK-END**

The MySQL Database Server is very fast, reliable, and easy to use. If that is what we are looking for, we should give it a try. MySQL Server also has a practical set of features developed in close cooperation with users. MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Though under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

**Technical Features of MYSQL Server**

The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backend, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs). MySQL Server provides a multi-threaded library which you can link into your application to get a smaller, faster, easier-to-manage product. There is a large amount of contributed MySQL software available. It is very likely that you will find that your favorite application or language already supports the MySQL Database Server.

**Database Features**

The following list describes some of the important characteristics of the MySQL Database Software.

* Written in C and C++.
* Tested with a broad range of different compilers.
* Works on many different platforms.
* Uses GNU Automake ,Autoconf, and Libtool for portability.
* APIs for C, C++, Eiffel, Java, Perl, PHP, Python, Ruby, and Tcl are available.
* Fully multi-threaded using kernel threads. This means it can easily use multiple CPUs if they are available.
* Provides transactional and non-transactional storage engines.
* Uses very fast B-tree disk tables (MyISAM) with index compression.
* Relatively easy to add another storage engine. This is useful if we want to add an SQL interface to an in-house database.
* A very fast thread-based memory allocation system.
* Very fast joins using an optimized one-sweep multi-join.
* In-memory hash tables which are used as temporary tables.
* SQL functions are implemented using a highly optimized class library and should be as fast as possible. Usually there is no memory allocation at all after query initialization.
* The MySQL code is tested with Purify (a commercial memory leakage detector) as well as with Valgrind, a GPL tool.
* The server is available as a separate program for use in a client/server networked environment. It is also available as a library that can be embedded (linked) into standalone applications. Such applications can be used in isolation or in environments where no network is available.
* Many column types: signed/unsigned integers 1, 2, 3, 4, and 8 bytes long, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET, ENUM, and OpenGIS geometry types.
* Fixed-length and variable-length records.

**Security**

* A privilege and password system that is very flexible and secure, and allows host-based verification. Passwords are secure because all password traffic is encrypted when you connect to a server.

**Scalability and Limits**

* Handles large databases. We use MySQL Server with databases that contain 50 million records. We also know of users that use MySQL Server with 60,000 tables and about 5,000,000,000 rows.
* Up to 32 indexes per table are allowed. Each index may consist of 1 to 16 columns or parts of columns. The maximum index width is 500 bytes (this may be changed when compiling MySQL Server). An index may use a prefix of a CHAR or VARCHAR column.

**Connectivity**

* Clients may connect to the MySQL server using TCP/IP sockets on any platform. On Windows systems in the NT family (NT, 2000, or XP), clients may connect using named pipes. On UNIX systems, clients may connect using UNIX domain socket files.
* The Connector/ODBC interface provides MySQL support for client programs that use ODBC (Open-Database-Connectivity) connections. For example, you can use MS Access to connect to your MySQL server. Clients may be run on Windows or UNIX. Connector/ODBC source is available. All ODBC 2.5 functions are supported, as are many others.
* The Connector/JDBC interface provides MySQL support for Java client programs that use JDBC connections. Clients may be run on Windows or UNIX. Connector/JDBC source is available.

**2. SYSTEM STUDY**

**2.1 Existing System**

In the Existing System, a service station has their own website in which user can book for their service request. Vehicle breakdowns are unpredictable and can be fatal, risk and stressful. In this situation they need the help of service providers but no other way to get the contact of nearby workstations or mechanics. This is one of the main drawback of the system. So, the proposed system is designed in that way to provide a solution for this situation.

**Drawbacks of Existing System**

* In efficient
* Hard to access the nearest workstation details
* Time consumption progress
* Limited access

**2.2 Proposed System**

Proposed system is designed to provide a helping hand to the people who are facing difficulties getting help when their vehicle breaks down. People can get the nearest workstation or mechanic details at anytime and anywhere. It provides full support to the people to rectify the vehicle related problems.

**Advantages of the Proposed System**

* Secure registration of user’s and mechanics.
* Easy access to the data.
* The new system is more user-friendly, reliable and flexible.
* 24x7 Support
* Saves the user time
* Mechanic can elaborate their business easily
* Good customer support

**3. SYSTEM DESIGN**

**3.1 INPUT DESIGN**

Input design is the process of converting an external user oriented description of the input system into a machine-oriented format. It is a part of overall system design, which requires very careful attention. In forms there are varieties of controls for designing user data entry screens. Designing the input is a significant part of while developing the system, because the input design may affect the feasibility of the system.

The major activities done with this are

* Data are collected from source
* Data are converted to computer acceptable form
* Validation of input data

When a data is to be entered the input data is validated. If the data is invalid, then the error message will be displayed on the screen.

**Registration**

New user can enter the system using this registration form. This form is used to get the details of the user such as name, address, contact and email.

**Location**

Location form is main form of this project. In this form it will gets the details of the location code and location name.

**Mechanic**

This form gets the details about various mechanics and workshops. It consists the information like workshop name, mechanic name and their contact details. This module allows the administrator to add, edit and delete mechanic information.

**3.2 OUTPUT DESIGN**

The output design was done so that the result of processing could be communicated to the users. The output defined with reference to the following. The frequency of the output, whether it was needed daily, weekly or monthly was defined. The accuracy requirements of the data were also defined. The system provides the following list of reports.

**Location wise Workshop**

Filters module is mainly used for search optimization purpose. It has various types of search facilities like area wise search, mechanic wise search etc.

**3.3 Database Design**

Database Design is crucial in the development of computer applications that maximize the effectiveness of the database, performance and overall quality of the application. Database design defines the method of data organization. The operational data is stored in entities. Relation between entities makes up data structure. Data structure is through the process of normalization.

Any information must be collected, accumulated, edited and should be retrieved. Therefore it should directly need files. The files should be properly designed and planned. Files are an important and crucial component for the successful performance of the system.

**4. SYSTEM DEVELOPMENT**

**4.1 DESCRIPTION OF MODULES**

**Admin**

Admin module is one of the key module of this project. Admin has unique authentication

**Registration**

New user can enter the system using this registration module. This module is used to get the details of the user such as name, address, contact and email.

**Mechanic**

This module has the details about various mechanics and workshops. It consists the information like workshop name, mechanic name and their contact details. This module allows the administrator to add, edit and delete mechanic information.

**Location**

This module is used to get the location details. In this module it includes the details of the location number, location Name, City, and landmark of the location.

**Filters**

Filters module is mainly used for search optimization purpose. It has various types of search facilities like area wise search, mechanic wise search etc.

**Chatbot**

This module is other way to find mechanic. Its also works like filter.

This module is used to get help from the admin team.

**5. SYSTEM TESTING**

**5.1 TESTING**

Testing is the process where the test data is prepared and used for testing the module individually and later the validation given for the fields. Actually testing is the state of implementation aimed at ensuring that the system works accurately and efficiently before the actual operation commences.

The software testing process commences one program is created and the documentation and related data structures are designed. Software testing is essential for correcting errors. Software testing is the process of checking whether the developed system is working according to the original objectives and requirements.

Testing is carried out after the development of the proposed system. The principle activity of system development is preparing the source code. In this system the source code is developed for each module separately. The source code is prepared for master files and they are compiled and corrected. Then the source code for the transaction files are prepared, compiled and corrected. Then the modules are combined and corrected as a whole module.

**TESTING STRATEGIES**

**Unit Testing**

Unit testing comprises the set of tests performed prior to integration of the unit into larger system. In unit testing every module are tested independently. Subroutines in every module are tested for the correct data flow and normal exit.

In this system each and every code files are tested after development. It verifies the work flow of the routine / module and the outputs are verified with the known data. After completing the each and every unit those are integrated under a page that connects all the forms according to user level such as database admin and user.

The first level of testing is called unit testing. This project involves several modules such as login, mechanic, area, filters are tested using unit testing.

**Integration Testing**

In the integration testing, the modules are joined together to make sure that they work as a group. Integration tests are performed incrementally by adding modules one by one to the main module and tested to make sure that it works properly after adding the next module.

**Validation Testing**

Validation can be defined in many ways but a simple definition is that validation succeeds when the software function in a manner that can be reasonably expected by the customer. Software validation is achieved through a series of tests demonstrate conformity with requirements. In certain places where the input has to be given, validation testing is done.

In this project key fields, numeric and date fields are validated to avoid improper data entry.

**5.2 SYSTEM SECURITY**

The security check includes user authentications. Only the users having the username and password can enter into the system. The username and password will get through the registration in the system. It also includes the option for changes in the account information.

To use the system user should have authorization for that. To get the system one should enter the username and password provided to them. If anyone types incorrect username or password, the error message will be displayed.

**6. SYSTEM IMPLEMENTATION**

It is an unrealistic attempt to test the system as a single unit. Thus the function of the system is tested and the functions are put together called modules are tested. Implementation refers to the process of converting a new or a revised system design into an operational one. The major activities in the implementation phase are

* Preparation of implementation
* Accessing equipment
* User or personal training
* Changeover plans

**Preparation of Implementation**

The implementation plan begins with preparing a plan for the implementation of the system. This plan was made regarding the equipment, resources available and how to test the activities. Thus a clean plan was prepared for all activities.

**Accessories Equipment**

In order to implement the proposed system the need of equipment has been assessed.

**User or Personal Training**

The following suggestions were given for user training,

* People involved in the organization has to be selected for implementation proposed system.
* Selected staff should be trained before the implementation plan.

**Change Over**

The following suggestions were made during the changeover plan,

* Change over has to be made carefully so as to minimize the problems that may arise from human error and machine malfunctioning.
* It is suggested to run parallel as the method of change over, here data from manual system, is run one the system.

**7. CONCLUSION**

This system provide the user friendly environment to the user. Using this system the user can easily gets the nearest workshop details and they can easily book their service. This system provide clear filtering techniques for data retrieval. The system is an integrated one, covering all aspects, which are required by the user. Updating of information becomes more effective and quicker. It is fully menu based with validation to check for accuracy of input data.

**8.** **FUTURE ENHANCEMENT**

This system is designed in user an integrated one, covering all aspects, which are required by the user. Updating of information becomes more effective and quicker. It is fully menu based with validation to check for accuracy of input data. Any further modifications can be made easily to extend the system to overcome the future enhancement in the concern. In future we can add the GPS based location services.

**9. BIBLIOGRAPHY**

**BOOK REFERENCE**

* Roger S.Presssman, “**SOFTWARE ENGINEERING**”, A Practitioners Approach, Tata-McGrew Hill Publication, 1998, First Edition.
* [Kevin Yank](https://www.cloudways.com/blog/best-php-books-to-help-you-learn/#novice-to-ninja),“**P**[**HP & MySQL Novice to Ninja – by Kevin Yank**](https://www.cloudways.com/blog/best-php-books-to-help-you-learn/#novice-to-ninja)”, Macmillan Computer Publishing.
* Luke Welling & Laura Thompson, [**PHP & MySQL Web Development**](https://www.cloudways.com/blog/best-php-books-to-help-you-learn/#luke-laura)**,**Addison Wesley publications.
* Ellias M Award “**SYSTEM ANALYSIS DESIGN”** Galgotia Publications Pvt. Ltd., II Edition 1996.

**WEB REFERENCE**

* <https://en.wikipedia.org/wiki/Flask_(web_framework)>
* <https://www.w3schools.com/python/default.asp>
* https://www.python.org/doc/essays/blurb/

**10. APPENDICES**

**10.1 DATAFLOW DIAGRAM**

**Level 0**

Search

Location

User

**Level 1**

User

Admin

AreaMaster

MechanicDet

Store mechanic details

Store registration details

Store Location details

Get mechanic details

Get registration details

LocationMaster

Registration

Location Based Search

**10.2 ENTITY RELATIONSHIP DIAGRAM**

LocationMaster

Loc\_ Code

Sign up

Loc\_ Code

Mechanic

**10.3 TABLE DESIGN**

Table Name : Location Master

Primary Key : lid

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Width** | **Description** |
| lid | Int |  | Location id |
| lname | Varchar | 50 | Location Name |
| lcity | Varchar | 70 | City |
| Landmark | Varchar | 70 | Landmark |

Table Name : Signup

Primary Key : Uid

Reference Key : lid

|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Width** | **Description** |
| Uid | Varchar | 7 | User id number |
| Username | Varchar | 30 | Username |
| Password | Varchar | 30 | Password |

Table Name : Mechanic

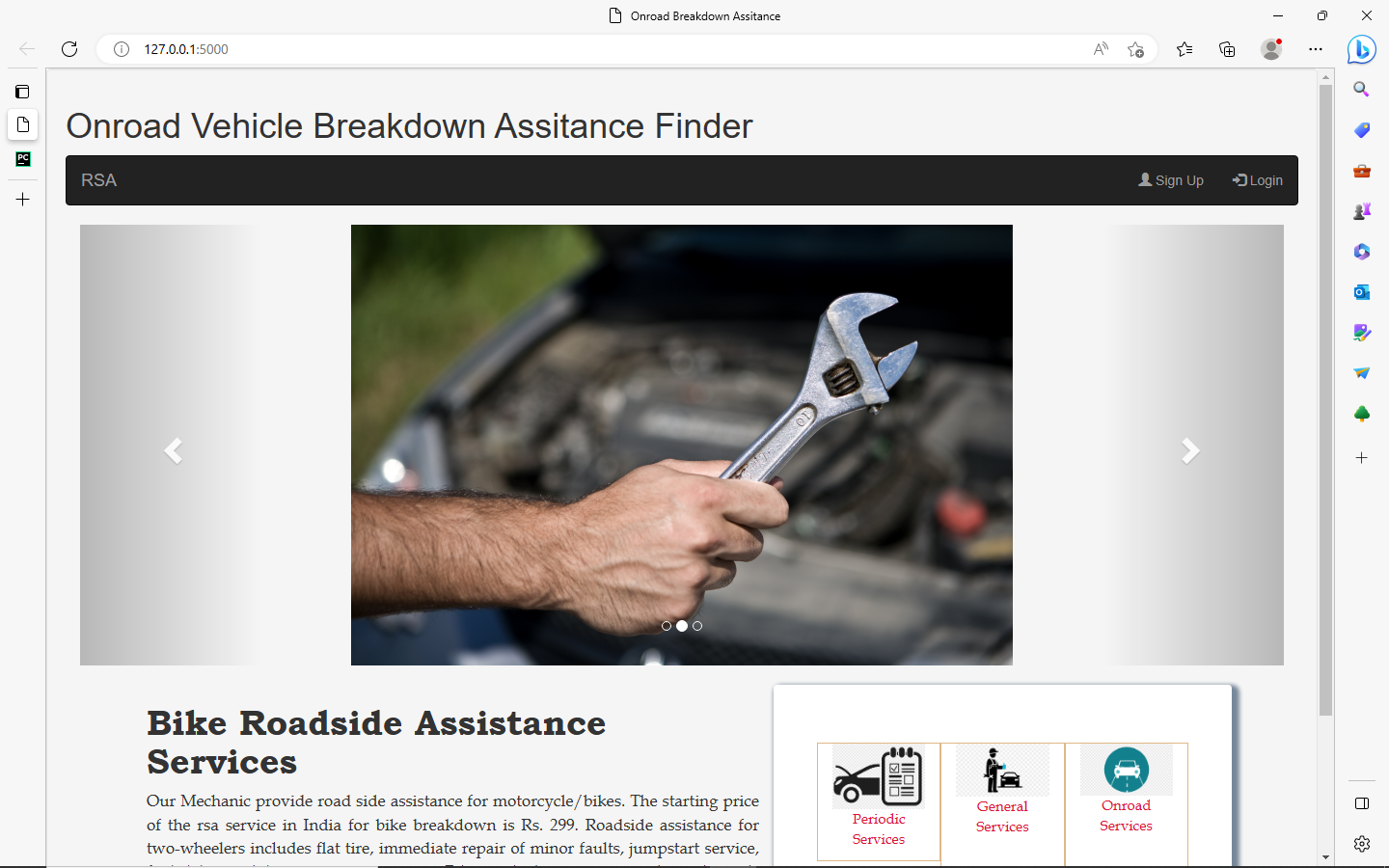
Primary Key : Mid

Reference Key : lid

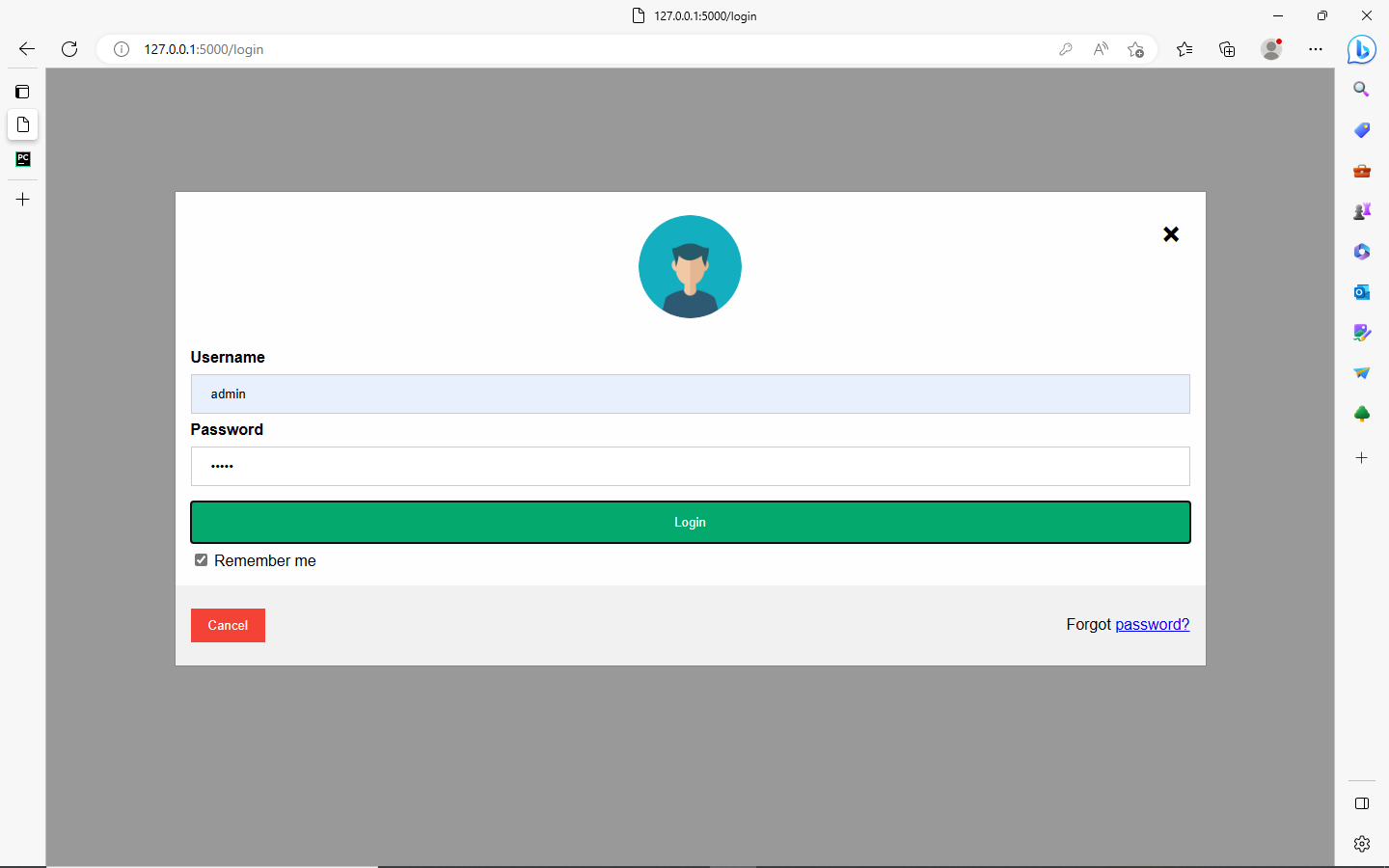
|  |  |  |  |
| --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Width** | **Description** |
| Mid | Int |  | Mechanic id |
| Mech\_Name | Varchar | 30 | Mechanic Name |
| Maddr | Varchar | 100 | Mechanic Address |
| MCont | int | 10 | Mechanic Contact |
| Wname | Varchar | 30 | Workshop Name |
| Wloc | Varchar | 30 | Workshop location |

**10.4 SAMPLE SCREENS**

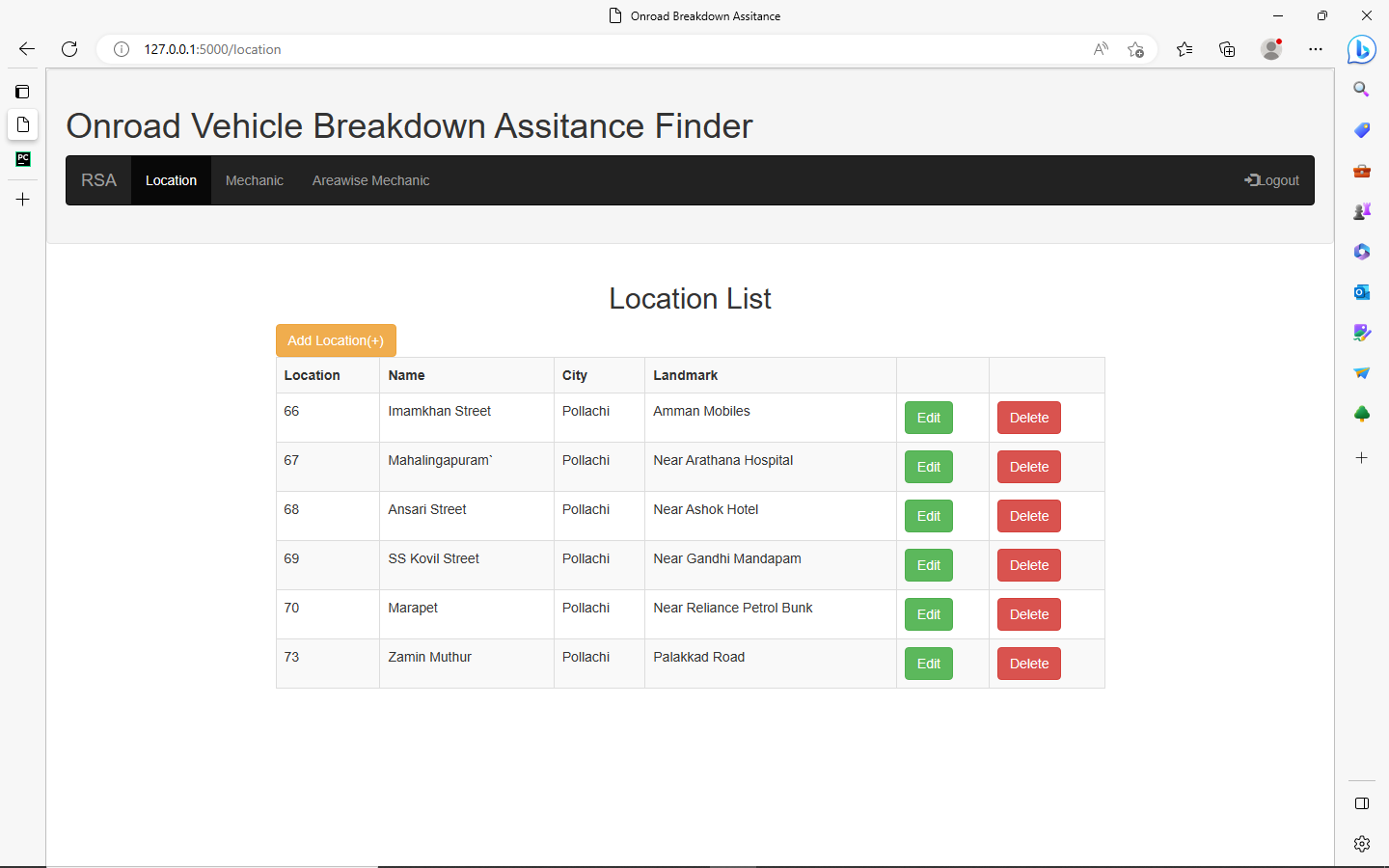
**HOME PAGE**



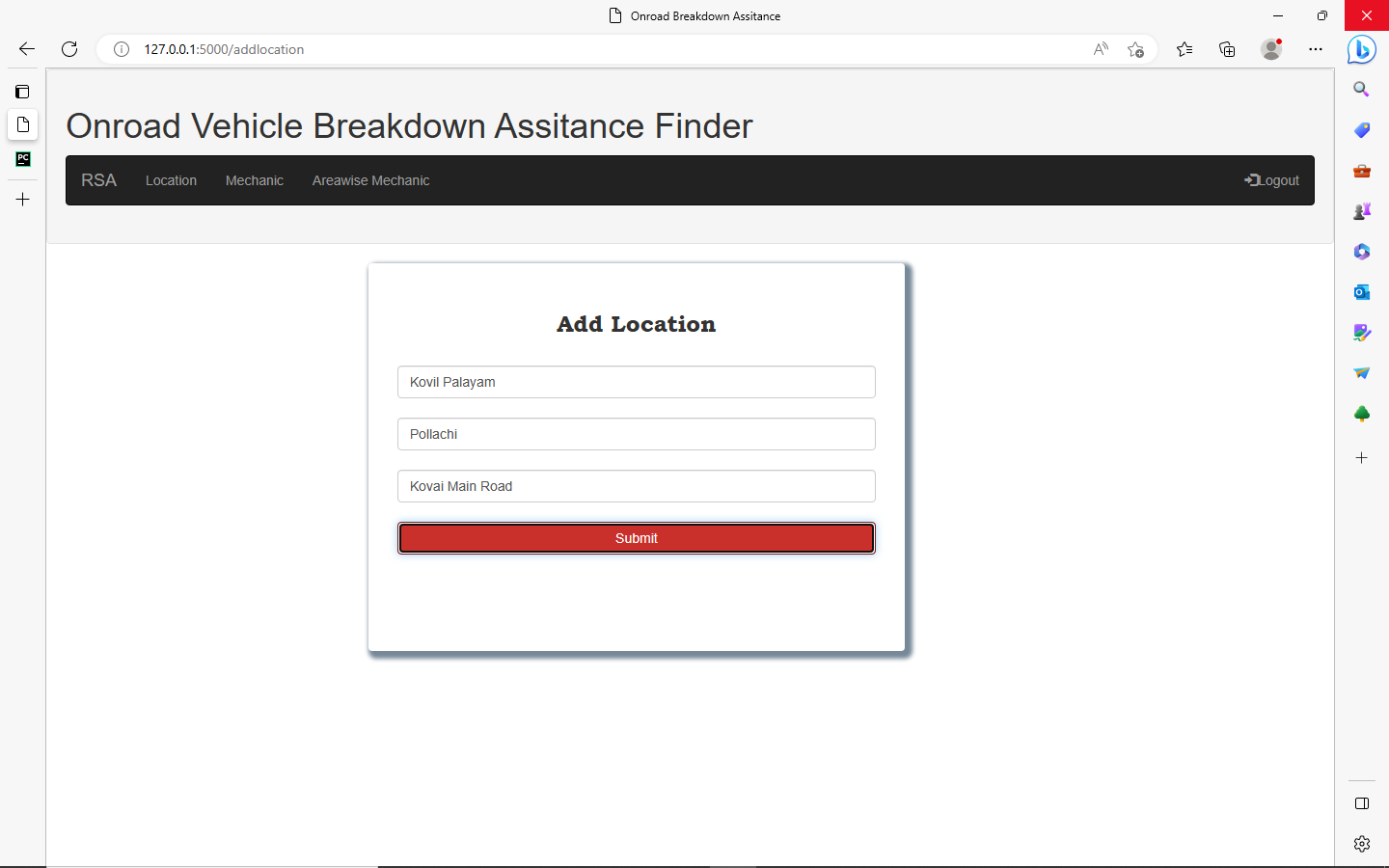
**Admin Login**



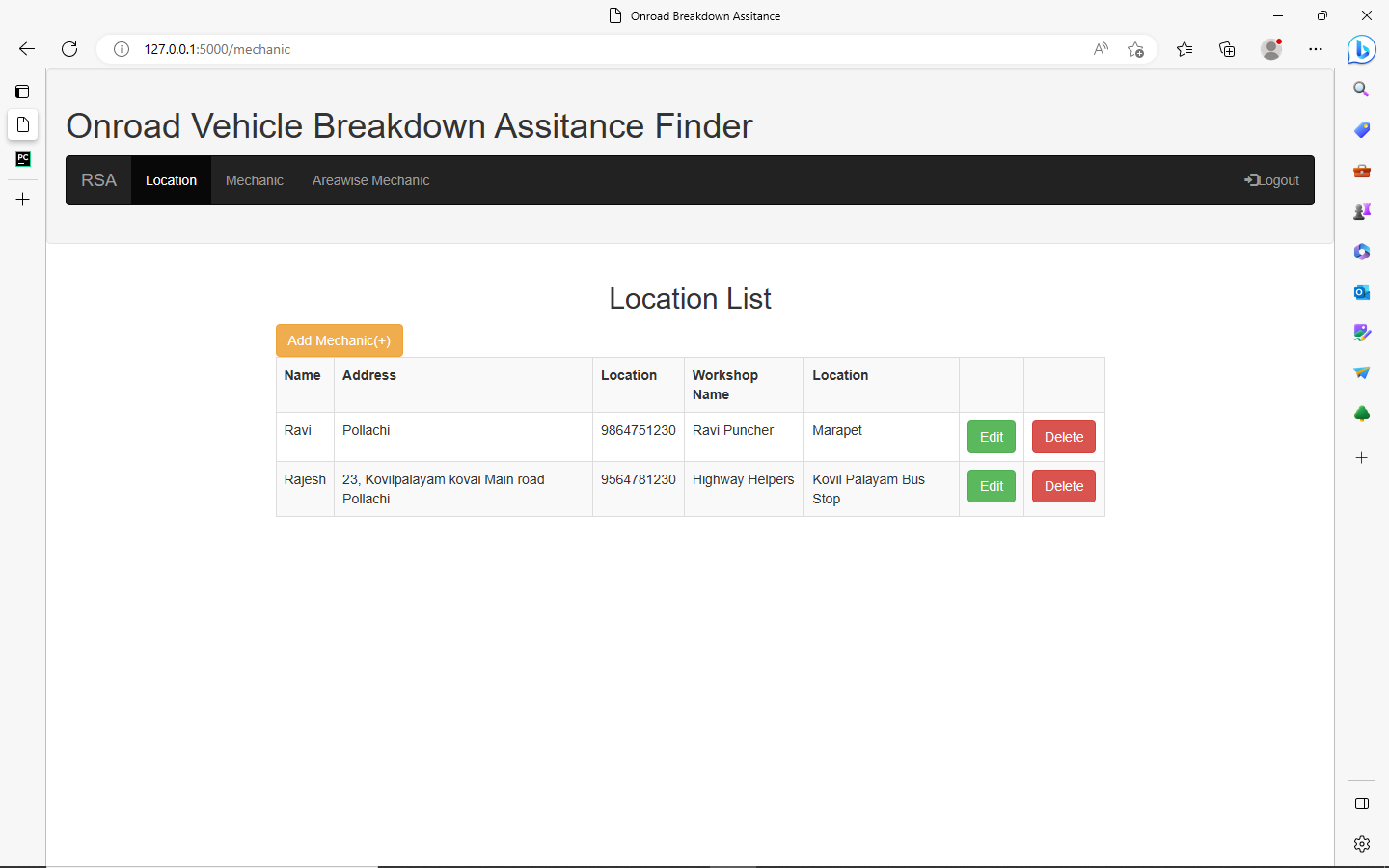
**Location List**



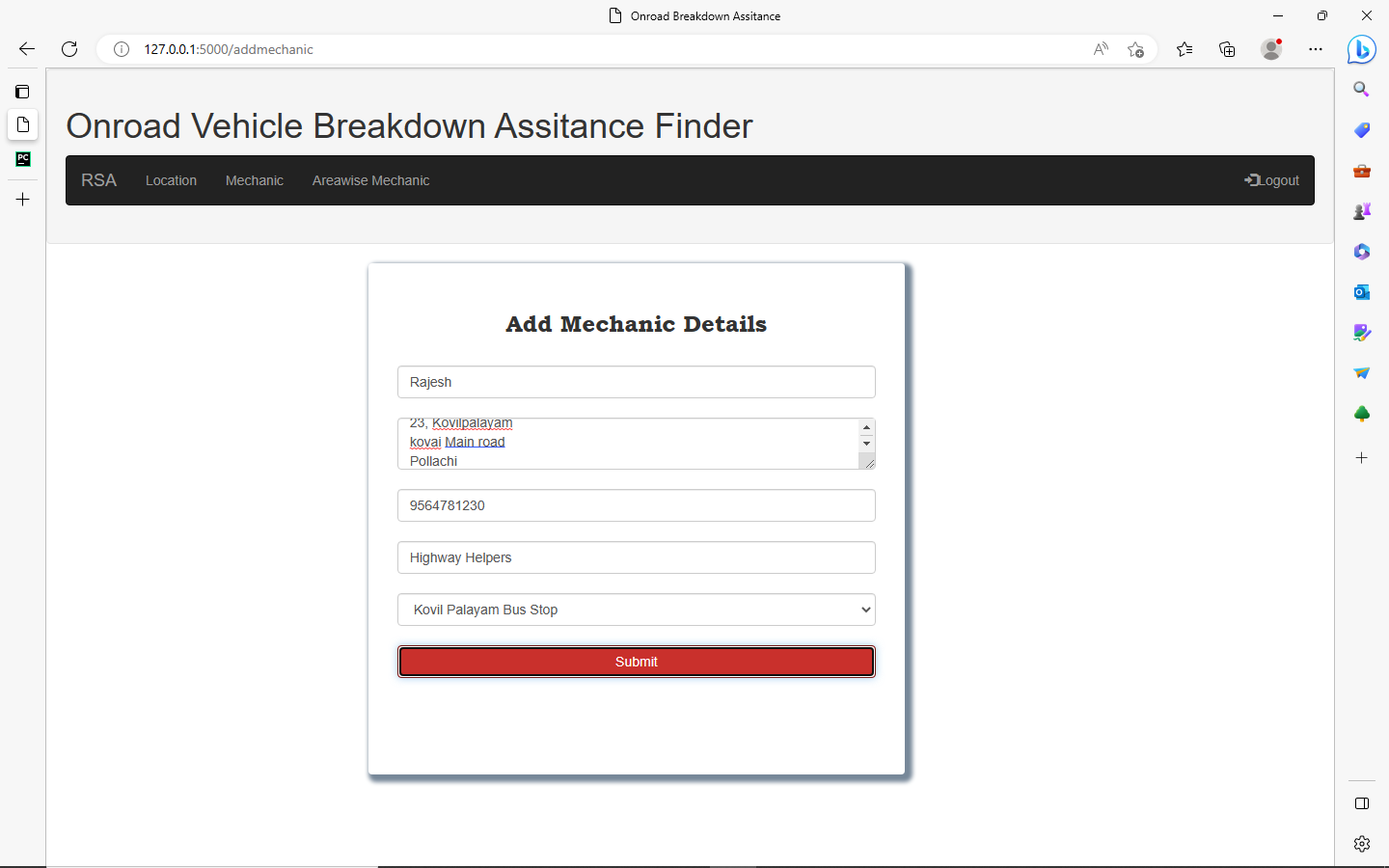
**Add Location**



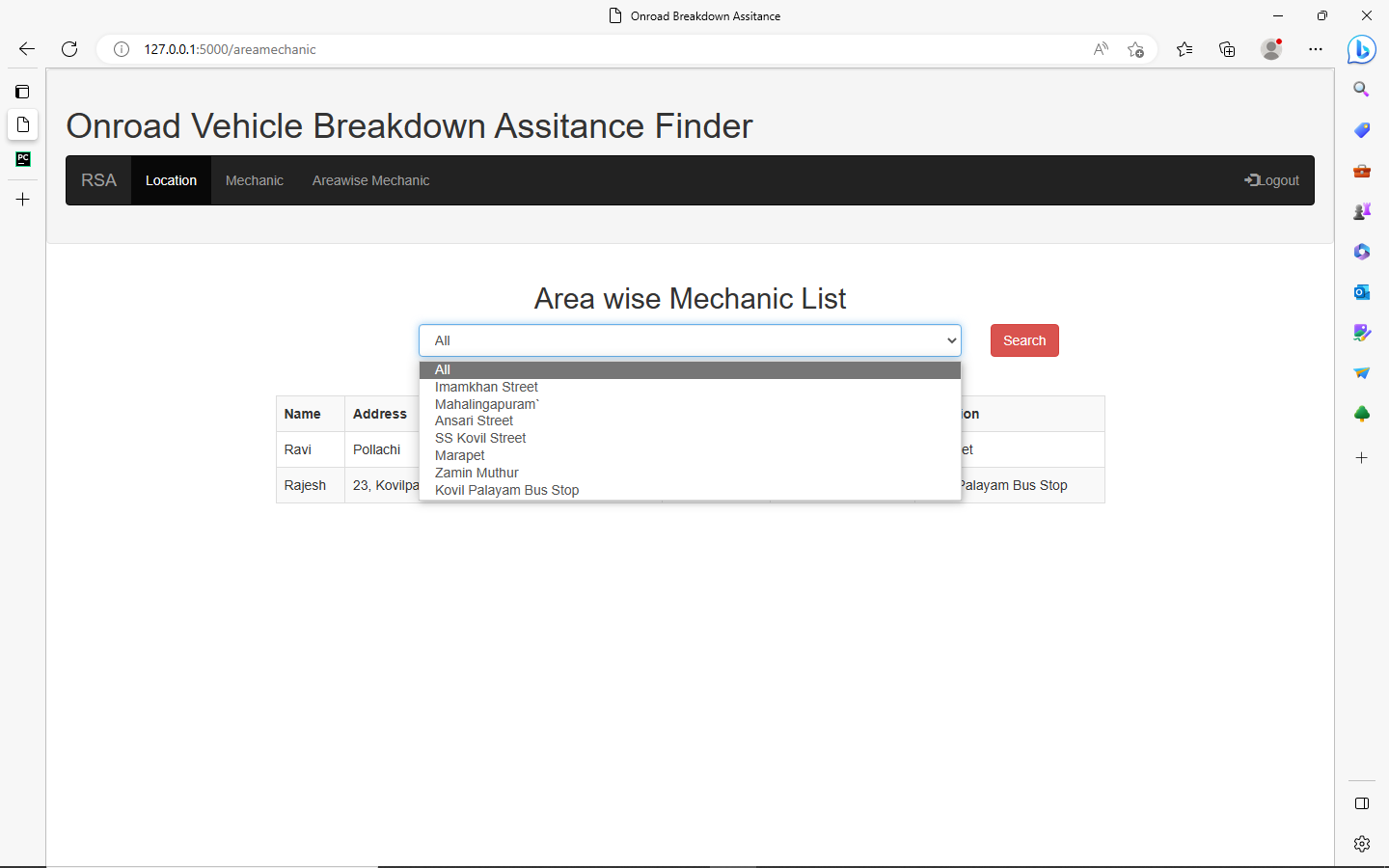
**Mechanic List**



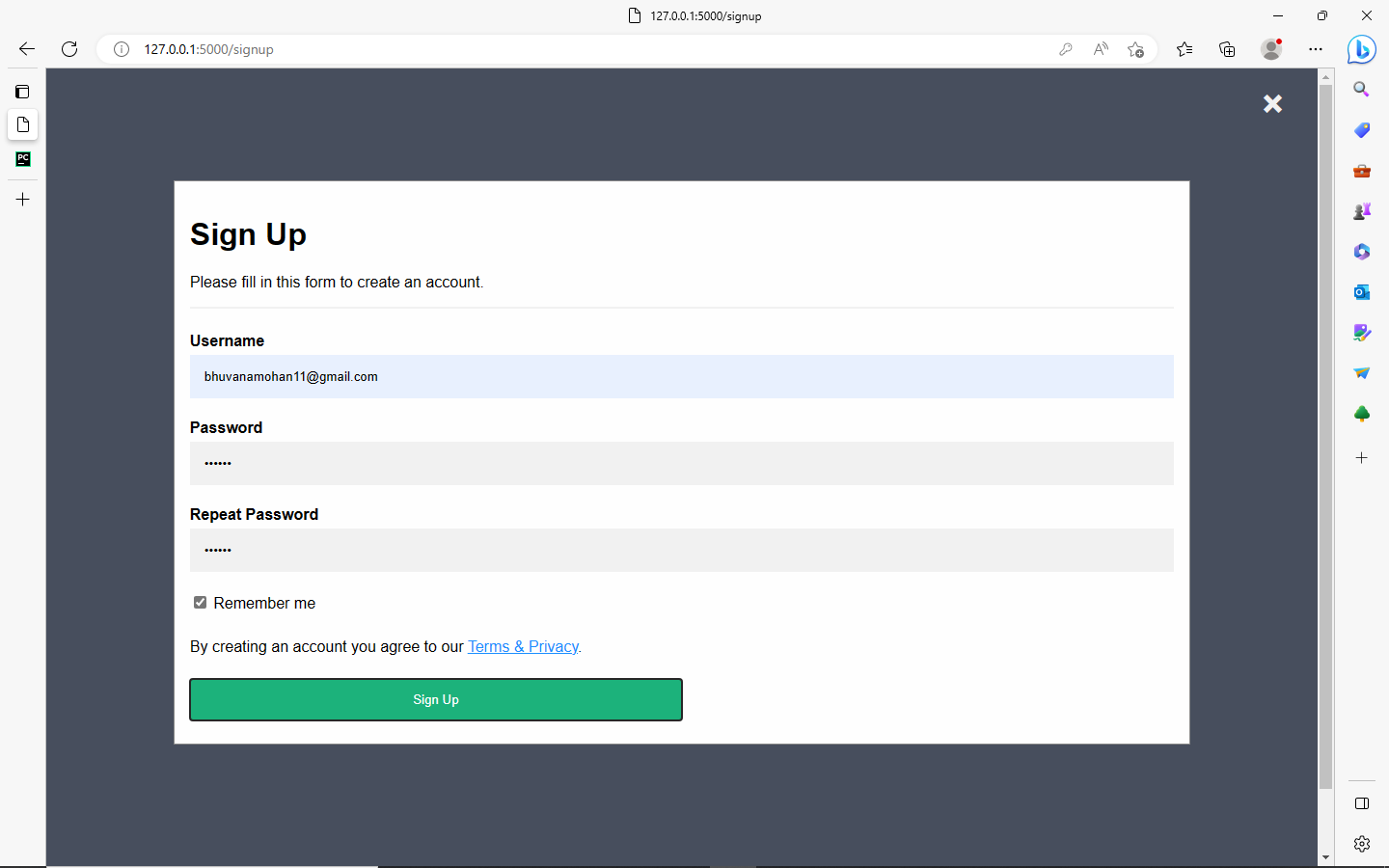
**Add Mechanic**



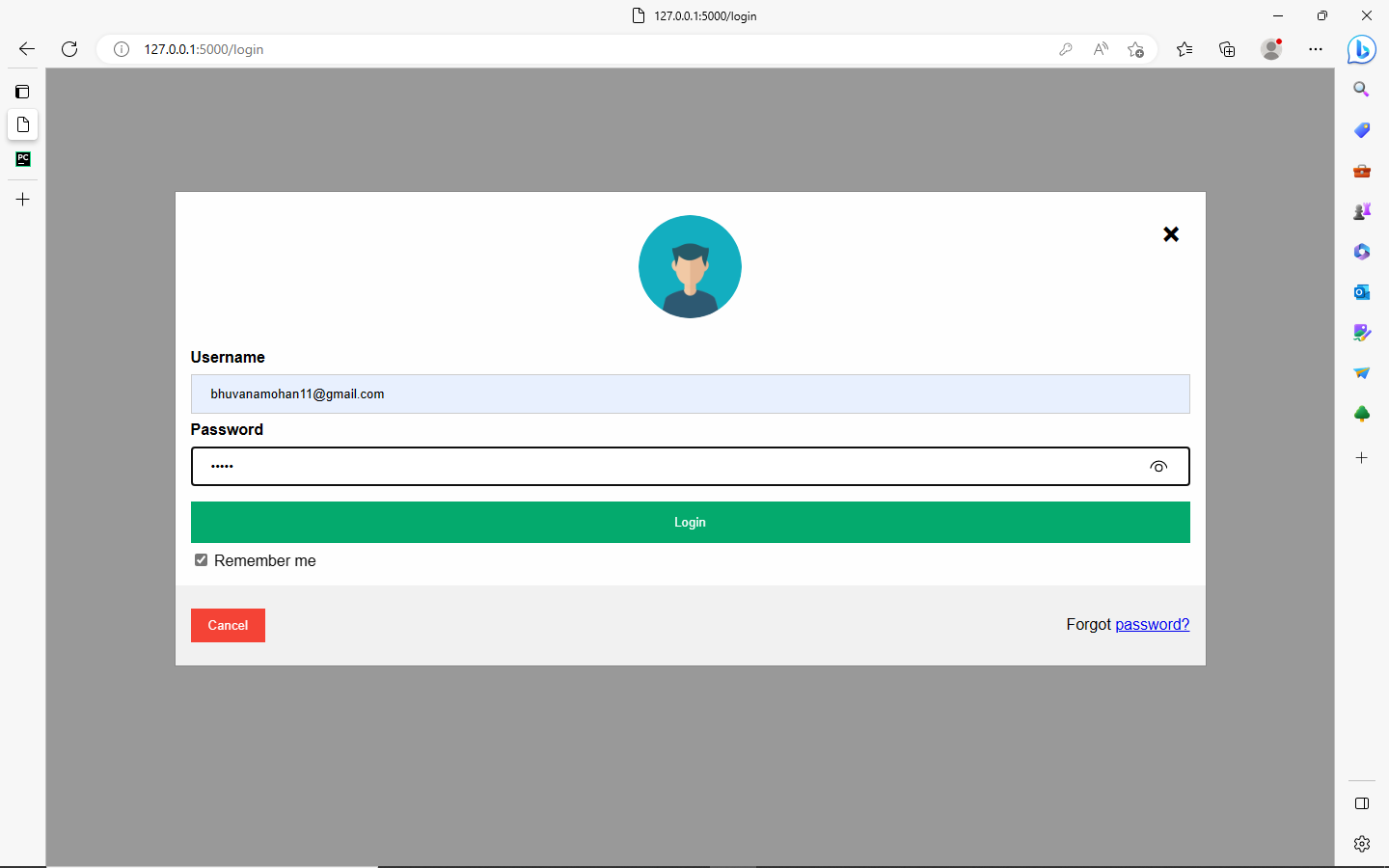
**Area wise Mechanic List**



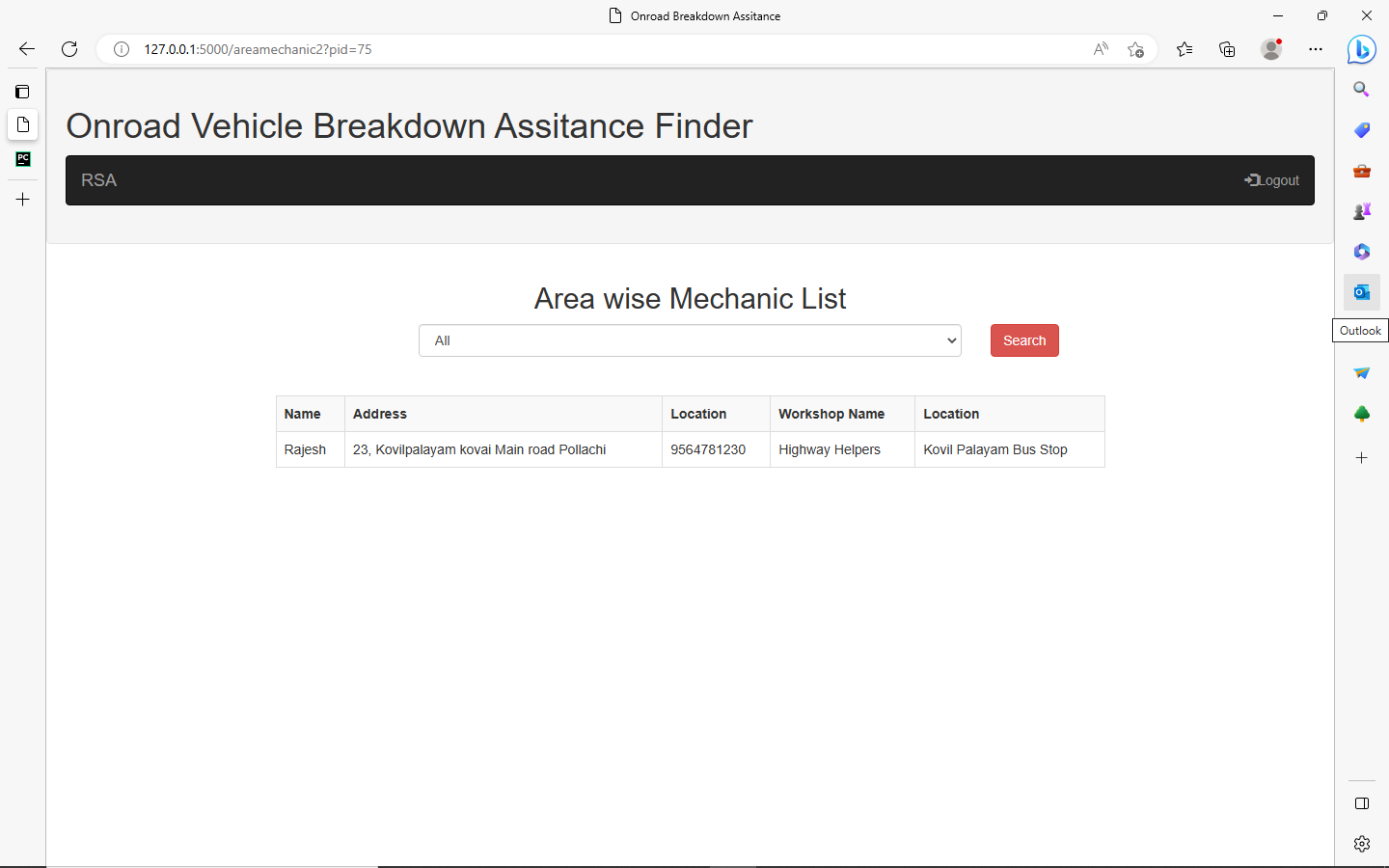
**User Signup**



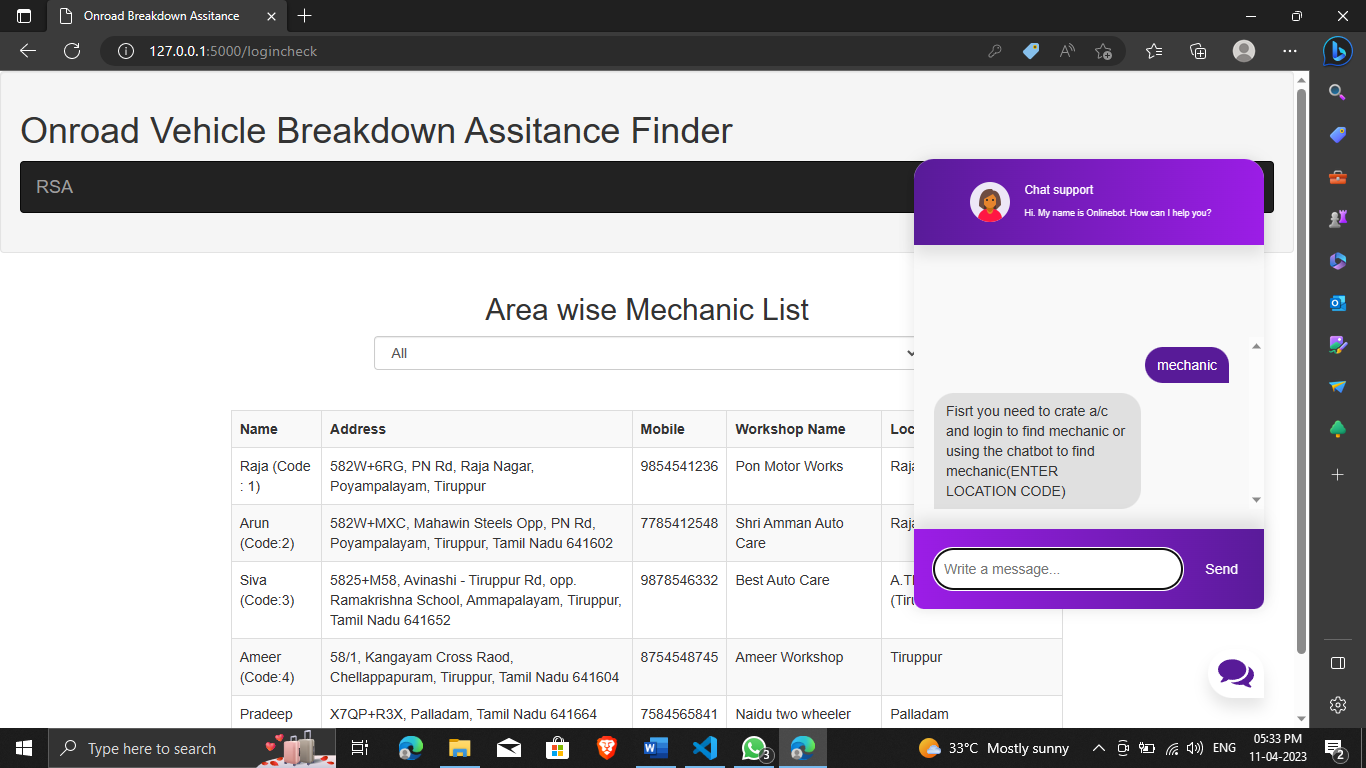
**Login**



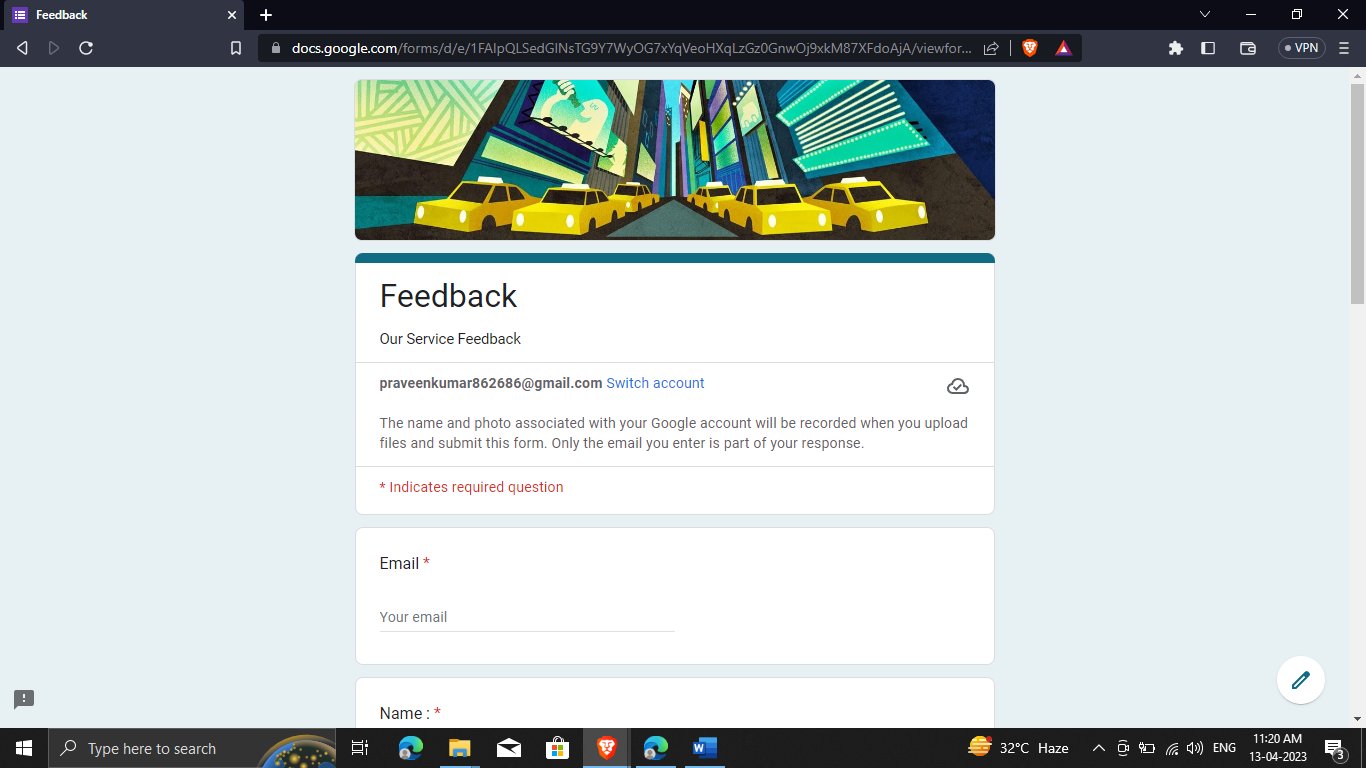
**Search Mechanic**

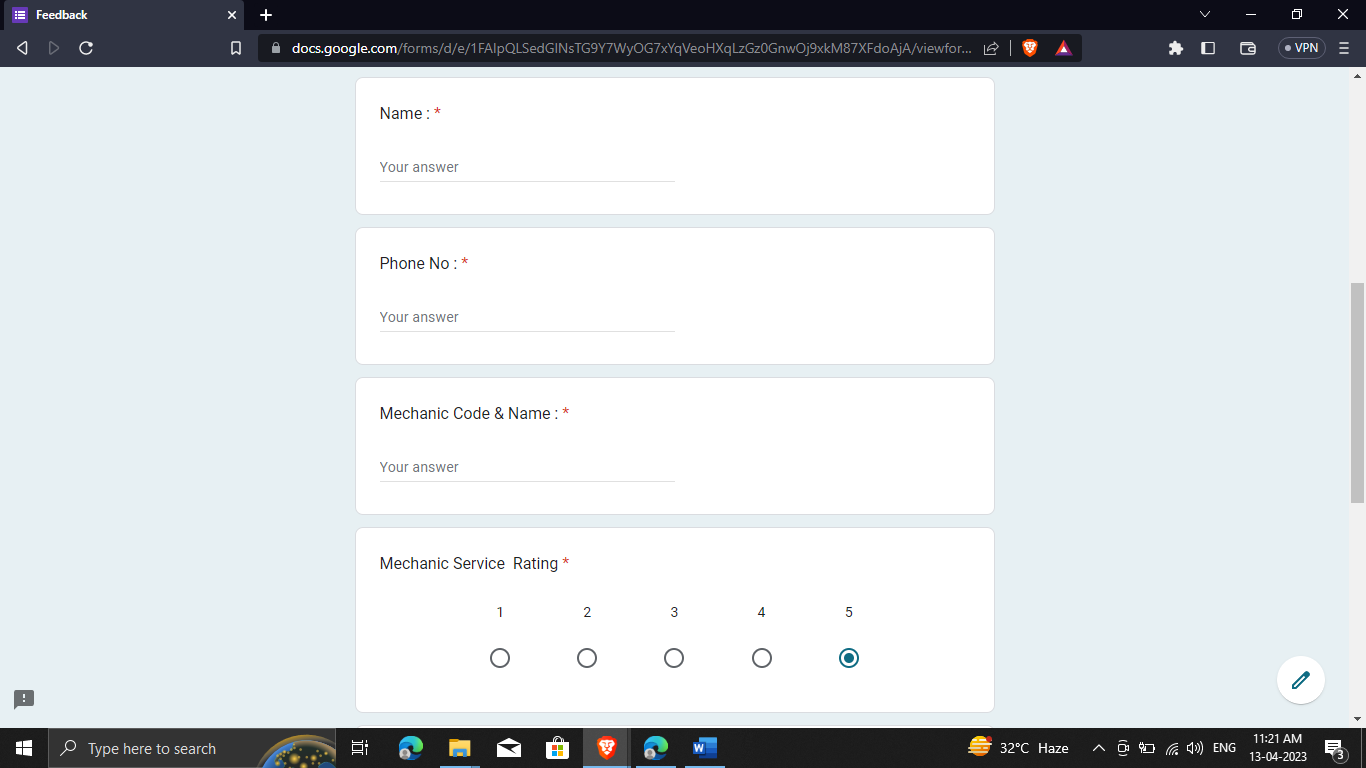


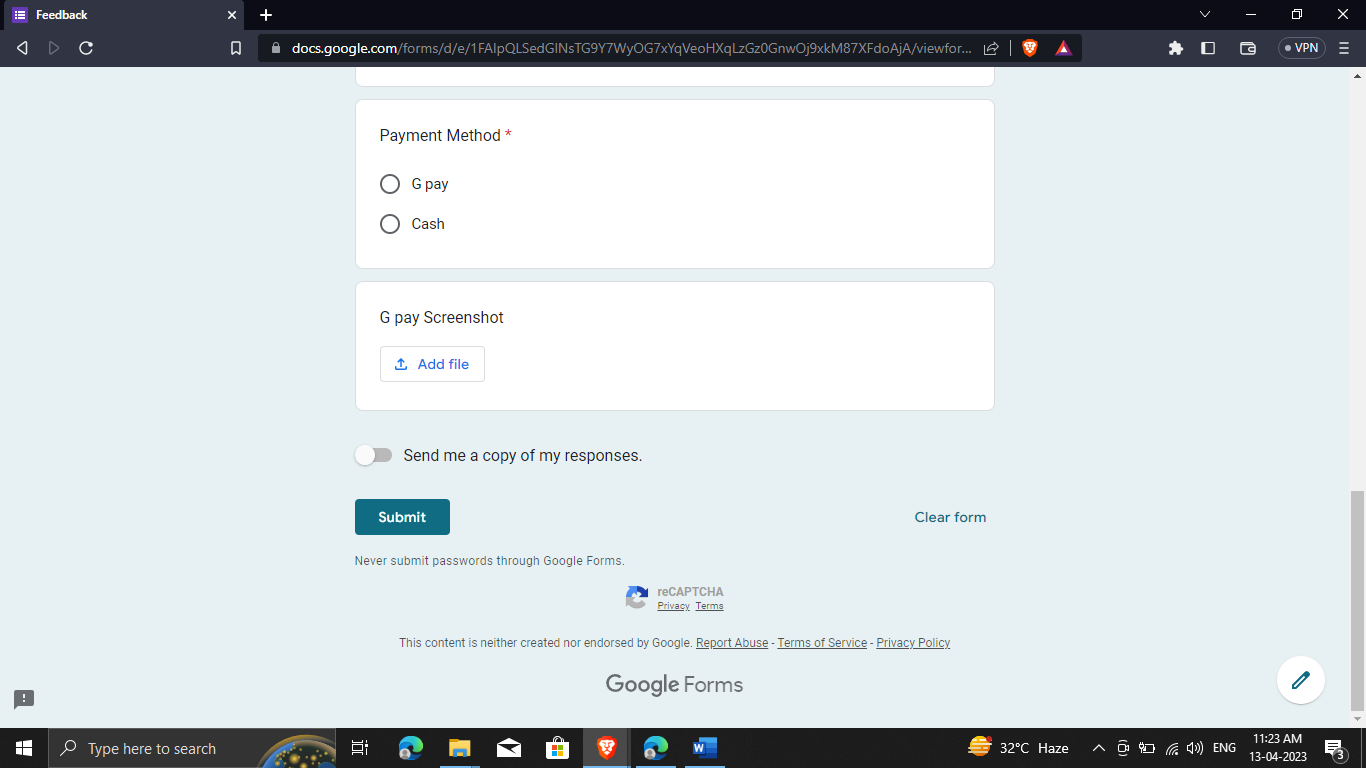
**Using Chatbot**



**Feedback form**







**Sample Coding**

<!DOCTYPE html>  
<html lang="en">  
<link rel="stylesheet" href="style.css">

<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <title>Onroad Breakdown Assitance </title>  
 <meta charset="utf-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js"></script>  
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.3/jquery.min.js"></script>  
 <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>  
</head>  
<body>  
<div class="well">  
 <h1>Onroad Vehicle Breakdown Assitance Finder </h1>  
<nav class="navbar navbar-inverse">  
 <div class="container-fluid">  
 <div class="navbar-header">  
 <a class="navbar-brand" href="#">RSA</a>  
 </div>  
 <ul class="nav navbar-nav">  
 <li><a href="{{ url\_for('location') }}">Location</a></li>  
 <li><a href="{{ url\_for('mechanic') }}">Mechanic</a></li>  
 <li><a href="{{ url\_for('areamechanic') }}">Areawise Mechanic</a></li>  
 </ul>  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="{{url\_for('index') }}"><span class="glyphicon glyphicon-log-in"></span>Logout</a></li>  
 </ul>  
 </div>  
</nav>  
</div>  
  
  
  
<div class="col-md-3"></div>  
<form action="{{ url\_for('insert') }}" method="post">  
 <div class="col-md-5" align="left" style="border-radius:4px;box-shadow:lightslategray 4px 4px 4px 4px; padding: 30px;padding-bottom:100px;background-color: white;">  
  
 <h3 align="center" style="font-family: 'Bookman Old Style'">Add Location</h3><br>  
 <input type="text" class="form-control" name="name" placeholder="Location Name">  
 <br>  
 <input type="text" class="form-control" name="city" placeholder="City">  
 <br>  
 <input type="text" class="form-control" name="Loc" placeholder="Nearest Landmark "><br>

<center><button type="submit" class="btn btn-danger form-control

">Submit</button></center>  
  
 </div>

<div class="container">

<div class="chatbox">

<div class="chatbox\_\_support">

<div class="chatbox\_\_header">

<div class="chatbox\_\_image--header">

<img src="https://img.icons8.com/color/48/000000/circled-user-female-skin-type-5--v1.png" alt="image">

</div>

<div class="chatbox\_\_content--header">

<h4 class="chatbox\_\_heading--header">Chat support</h4>

<p class="chatbox\_\_description--header">Hi. My name is Sam. How can I help you?</p>

</div>

</div>

<div class="chatbox\_\_messages">

<div></div>

</div>

<div class="chatbox\_\_footer">

<input type="text" placeholder="Write a message...">

<button class="chatbox\_\_send--footer send\_\_button">Send</button>

</div>

</div>

<div class="chatbox\_\_button">

<button><img src="./images/chatbox-icon.svg" /></button>

</div>

</div>

</div>

<script src="./app.js"></script>

</form>  
</body>  
</html>

from urllib.parse import urlparse

import MySQLdb

from flask import Flask, render\_template, request, redirect, url\_for, jsonify

from flask\_mysqldb import MySQL

from chat import get\_response

app = Flask(\_\_name\_\_)

app.config['MYSQL\_HOST'] = 'localhost'

app.config['MYSQL\_USER'] = 'praveen'

app.config['MYSQL\_PASSWORD'] = 'praveen'

app.config['MYSQL\_DB'] = 'breakdown'

mysql = MySQL(app)

@app.route('/')

def index(): # put application's code here

return render\_template('index.html')

@app.route('/login')

def login(): # put application's code here

return render\_template('login.html')

@app.route('/location')

def location():

cur = mysql.connection.cursor()

cur.execute("SELECT \* from location")

data = cur.fetchall()

cur.close()

return render\_template('Locationlist.html', place=data)

@app.route('/logincheck', methods=['POST', 'GET'])

def logincheck():

if request.method == "POST":

name = request.form['uname']

psw = request.form['psw']

if name=="admin":

cur = mysql.connection.cursor()

query = "SELECT \* from signup where username=%s and password = %s"

vals = (name, psw,)

cur.execute(query, vals)

data = cur.fetchall()

cur.close()

if len(data)>0:

cur = mysql.connection.cursor()

cur.execute("SELECT \* from mechanic,location where mechanic.wloc=location.lid")

mechdata = cur.fetchall()

cur.close()

cur = mysql.connection.cursor()

cur.execute("SELECT \* from location")

mecharea = cur.fetchall()

cur.close()

return render\_template('Locationlist.html', mech=mechdata, mechloc=mecharea, userdet=data)

else:

return ("invalid login")

else:

cur = mysql.connection.cursor()

query="SELECT \* from signup where username=%s and password = %s"

vals=(name,psw,)

cur.execute(query,vals)

data = cur.fetchall()

cur.close()

if len(data) > 0:

cur = mysql.connection.cursor()

cur.execute("SELECT \* from mechanic,location where mechanic.wloc=location.lid")

mechdata = cur.fetchall()

cur.close()

cur = mysql.connection.cursor()

cur.execute("SELECT \* from location")

mecharea = cur.fetchall()

cur.close()

return render\_template('userareawisemechaniclist.html', mech=mechdata, mechloc=mecharea, userdet=data)

else:

return ("invalid login")

@app.route('/logincheck1', methods=['POST', 'GET'])

def logincheck1():

if request.method == "POST":

name = request.form['uname']

psw = request.form['psw']

cur = mysql.connection.cursor()

query = "SELECT \* from signup where username=%s and password = %s"

vals = (name, psw,)

cur.execute(query, vals)

dataval = cur.fetchall()

cur.close()

if len(dataval)>0:

return str(("Total rows are", len(dataval)))

else:

return ("Not valid")

@app.route('/mechanic')

def mechanic():

cur = mysql.connection.cursor()

cur.execute("SELECT \* from mechanic,location where mechanic.wloc=location.lid")

mechdata = cur.fetchall()

cur.close()

return render\_template('mechaniclist.html', mech=mechdata)

@app.route('/areamechanic')

def areamechanic():

cur = mysql.connection.cursor()

cur.execute("SELECT \* from mechanic,location where mechanic.wloc=location.lid")

mechdata = cur.fetchall()

cur.close()

cur = mysql.connection.cursor()

cur.execute("SELECT \* from location")

mecharea = cur.fetchall()

cur.close()

return render\_template('areawisemechaniclist.html', mech=mechdata, mechloc=mecharea)

@app.route('/areamechanic1')

def areamechanic1():

mechid = int(request.args.get("pid"))

if mechid == 1:

cur = mysql.connection.cursor()

cur.execute("SELECT \* from mechanic,location where mechanic.wloc=location.lid")

mechdata = cur.fetchall()

cur.close()

else:

cur = mysql.connection.cursor()

qrstr= "SELECT \* from mechanic,location where mechanic.wloc=location.lid and mechanic.wloc=%s"

locid=(mechid,)

cur.execute(qrstr, locid)

mechdata = cur.fetchall()

cur.close()

cur = mysql.connection.cursor()

cur.execute("SELECT \* from location")

mecharea = cur.fetchall()

cur.close()

return render\_template('areawisemechaniclist.html', mech=mechdata, mechloc=mecharea)

@app.route('/areamechanic2')

def areamechanic2():

mechid = int(request.args.get("pid"))

if mechid == 1:

cur = mysql.connection.cursor()

cur.execute("SELECT \* from mechanic,location where mechanic.wloc=location.lid")

mechdata = cur.fetchall()

cur.close()

else:

cur = mysql.connection.cursor()

qrstr= "SELECT \* from mechanic,location where mechanic.wloc=location.lid and mechanic.wloc=%s"

locid=(mechid,)

cur.execute(qrstr, locid)

mechdata = cur.fetchall()

cur.close()

cur = mysql.connection.cursor()

cur.execute("SELECT \* from location")

mecharea = cur.fetchall()

cur.close()

return render\_template('userareawisemechaniclist.html', mech=mechdata, mechloc=mecharea)

@app.route('/addlocation')

def addlocation(): # put application's code here

return render\_template('Location.html')

@app.route('/signup')

def signup(): # put application's code here

return render\_template('signup.html')

@app.route('/signupinsert', methods = ['POST', 'GET'])

def signupinsert(): # put application's code here

if request.method == "POST":

name = request.form['email']

psw = request.form['psw']

cur = mysql.connection.cursor()

cur.execute("insert into signup (username,password) VALUES (%s,%s)", (name, psw))

mysql.connection.commit()

print("inserted")

return redirect(url\_for('index'))

@app.route('/addmechanic')

def addmechanic(): # put application's code here

cur = mysql.connection.cursor()

cur.execute("SELECT \* from location")

mechloc = cur.fetchall()

cur.close()

return render\_template('mechanic.html', mechloc=mechloc)

@app.route('/editlocation')

def editlocation(): # put application's code here

locid=int(request.args.get("id"))

cur = mysql.connection.cursor()

sql = "SELECT \* from location where lid = %s"

adr = (locid,)

cur.execute(sql, adr)

data = cur.fetchall()

cur.close()

return render\_template('locationedit.html', placeedit=data)

@app.route('/editmech')

def editmech(): # put application's code here

mechid = int(request.args.get("mid"))

cur = mysql.connection.cursor()

sql = "SELECT \* from mechanic,location where mechanic.wloc=location.lid and mechid = %s"

adr = (mechid,)

cur.execute(sql, adr)

data = cur.fetchall()

cur.close()

cur = mysql.connection.cursor()

cur.execute("SELECT \* from location")

newdata = cur.fetchall()

cur.close()

return render\_template('editmech.html', mechedit=data, seldata=newdata)

@app.route('/update', methods=['POST', 'GET'])

def update():

if request.method == "POST":

lid = request.form['lid']

name = request.form['locname']

city = request.form['city']

loc = request.form['Loc']

cur = mysql.connection.cursor()

cur.execute(""" UPDATE location

set lname=%s, lcity=%s, landmark=%s

where lid=%s

""", (name, city, loc, lid))

mysql.connection.commit()

return redirect(url\_for('location'))

@app.route('/mechupdate', methods=['POST', 'GET'])

def mechupdate():

if request.method == "POST":

mid = request.form['mechid']

mname = request.form['mname']

maddr = request.form['maddr']

mcont = request.form['mcont']

wname = request.form['wname']

wloc = request.form['wloc']

cur = mysql.connection.cursor()

cur.execute(""" UPDATE mechanic

set mname=%s, maddr=%s, mcont=%s, wname=%s, wloc=%s

where mechid=%s

""", (mname, maddr, mcont, wname, wloc,mid))

mysql.connection.commit()

return redirect(url\_for('mechanic'))

@app.route('/delete', methods=['POST', 'GET'])

def delete():

id\_data = int(request.args.get("id\_data"))

cur = mysql.connection.cursor()

qstr = "DELETE FROM location where lid = %s"

ddata=(id\_data,)

cur.execute(qstr, ddata,)

mysql.connection.commit()

return redirect(url\_for('location'))

@app.route('/mechdelete', methods=['POST', 'GET'])

def mechdelete():

id\_data = int(request.args.get("id\_mechdata"))

cur = mysql.connection.cursor()

qstr = "DELETE FROM mechanic where mechid = %s"

ddata=(id\_data,)

cur.execute(qstr, ddata,)

mysql.connection.commit()

return redirect(url\_for('mechanic'))

@app.route('/insert', methods=['POST'])

def insert():

if request.method == "POST":

name = request.form['name']

city = request.form['city']

loc = request.form['Loc']

cur = mysql.connection.cursor()

cur.execute("insert into location (lname,lcity,landmark) VALUES (%s,%s,%s)", (name, city, loc))

mysql.connection.commit()

print("inserted")

return redirect(url\_for('location'))

@app.route('/mechinsert', methods=['POST'])

def mechinsert():

if request.method == "POST":

mname = request.form['mname']

maddr = request.form['maddr']

mcont = request.form['mcont']

wname = request.form['wname']

loc = request.form['selloc']

cur = mysql.connection.cursor()

cur.execute("insert into mechanic (mname,maddr, mcont,wname,wloc) VALUES (%s,%s,%s,%s,%s)", (mname, maddr, mcont, wname, loc))

mysql.connection.commit()

print("inserted")

return redirect(url\_for('mechanic'))

@app.post("/predict")

def predict():

text = request.get\_json().get("message")

# TODO : check if text is valid

response = get\_response(text)

message = {"answer": response}

return jsonify(message)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

import numpy as np

import random

import json

import torch

import torch.nn as nn

from torch.utils.data import Dataset, DataLoader

from nltk\_utils import bag\_of\_words, tokenize, stem

from model import NeuralNet

with open('intents.json', 'r') as f:

intents = json.load(f)

all\_words = []

tags = []

xy = []

# loop through each sentence in our intents patterns

for intent in intents['intents']:

tag = intent['tag']

# add to tag list

tags.append(tag)

for pattern in intent['patterns']:

# tokenize each word in the sentence

w = tokenize(pattern)

# add to our words list

all\_words.extend(w)

# add to xy pair

xy.append((w, tag))

# stem and lower each word

ignore\_words = ['?', '.', '!']

all\_words = [stem(w) for w in all\_words if w not in ignore\_words]

# remove duplicates and sort

all\_words = sorted(set(all\_words))

tags = sorted(set(tags))

print(len(xy), "patterns")

print(len(tags), "tags:", tags)

print(len(all\_words), "unique stemmed words:", all\_words)

# create training data

X\_train = []

y\_train = []

for (pattern\_sentence, tag) in xy:

# X: bag of words for each pattern\_sentence

bag = bag\_of\_words(pattern\_sentence, all\_words)

X\_train.append(bag)

# y: PyTorch CrossEntropyLoss needs only class labels, not one-hot

label = tags.index(tag)

y\_train.append(label)

X\_train = np.array(X\_train)

y\_train = np.array(y\_train)

# Hyper-parameters

num\_epochs = 1000

batch\_size = 8

learning\_rate = 0.001

input\_size = len(X\_train[0])

hidden\_size = 8

output\_size = len(tags)

print(input\_size, output\_size)

class ChatDataset(Dataset):

def \_\_init\_\_(self):

self.n\_samples = len(X\_train)

self.x\_data = X\_train

self.y\_data = y\_train

# support indexing such that dataset[i] can be used to get i-th sample

def \_\_getitem\_\_(self, index):

return self.x\_data[index], self.y\_data[index]

# we can call len(dataset) to return the size

def \_\_len\_\_(self):

return self.n\_samples

dataset = ChatDataset()

train\_loader = DataLoader(dataset=dataset,

batch\_size=batch\_size,

shuffle=True,

num\_workers=0)

device = torch.device('cuda' if torch.cuda.is\_available() else 'cpu')

model = NeuralNet(input\_size, hidden\_size, output\_size).to(device)

# Loss and optimizer

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=learning\_rate)

# Train the model

for epoch in range(num\_epochs):

for (words, labels) in train\_loader:

words = words.to(device)

labels = labels.to(dtype=torch.long).to(device)

# Forward pass

outputs = model(words)

# if y would be one-hot, we must apply

# labels = torch.max(labels, 1)[1]

loss = criterion(outputs, labels)

# Backward and optimize

optimizer.zero\_grad()

loss.backward()

optimizer.step()

if (epoch+1) % 100 == 0:

print (f'Epoch [{epoch+1}/{num\_epochs}], Loss: {loss.item():.4f}')

print(f'final loss: {loss.item():.4f}')

data = {

"model\_state": model.state\_dict(),

"input\_size": input\_size,

"hidden\_size": hidden\_size,

"output\_size": output\_size,

"all\_words": all\_words,

"tags": tags

}

FILE = "data.pth"

torch.save(data, FILE)

print(f'training complete. file saved to {FILE}')