ROAD MECH

A Mini Project Report

submitted by

SYAM KRISHNA P (KITM24MCA-2028)

To

The APJ Abdul Kalam Technological University
In partial fulfillment of the requirements for the award of the degree

of

Master of Computer Applications

Department of Computer Applications

KMCT INSTITUTE OF
TECHNOLOGY AND
MANAGEMENT
Kuttippuram, Malappuram - 679571

OCTOBER 2025

DECLARATION

I undersigned hereby declare that the project report ROAD MECH, submitted for partial fulfillment of the requirements for the award of degree of Master of ComputerApplications of the APJ Abdul Kalam Technological University, Kerala, is a bona fide work doneby me under supervision of Supervisor, Assistant Professor, Department of Computer Applications.This submission represents my ideas in my own words and where ideas or words of others have been included. I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresentedor fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis forthe award of any degree, diploma or similar title of any other University.

Place:Kuttippuram

Date:17/10/2025

DEPARTMENT OF COMPUTER APPLICATION
KMCT INSTITUTE OF TECHNOLOGY AND MANAGEMENT

CERTIFICATE

This is to certify that the report entitled **ROAD MECH** is a bona fiderecord of the Mini Project work during the year 2025-26 carried out by **SYAM KRISHNA P(KITM24MCA-2028)** submitted to the APJ Abdul Kalam Technological University, in partialfulfillment of the requirements for the award of the Master of Computer Applications, under my guidance and supervision. This report in any form has not been submitted to any other University or Institution for any purpose.

Internal Supervisor(s) External Supervisor(s)

Head of The Department

Acknowledgement

At the very outset I would like to thank the almighty's mercy towards me over the years. I wish to express my sincere thanks to my project coordinator, Ms. LAKSHMI R MENON head of the dept, Dept. of Master of Computer Applications who guided me for the successful completeness of this project. I also thank her for valuable suggestions, guidance, constant encouragement, boundless corporation, constructive comments and motivation extended to me for completion of this project work.

I would express my sincere thanks to my internal guide Ms. ASWATHY P NAIR guidanceto complete the project successfully.

I would like to express my sincere thanks to all the faculty members of Master of Computer Applications department for their support and valuable suggestion for doing the project work. Last but not least my graceful thanks to my parents, friends and also the persons who supported me directly and indirectly during the project.

SYAM KRISHNA P

(KITM24MCA-2028)

Abstract

RoadMech is a web-based application developed using the Django framework to the process of obtaining immediate roadside vehicle assistance in local areas. The platform serves three types of users: Admin, Mechanic/Service Center, and Vehicle Owner. The Admin manages the entire system, including mechanic approvals and service monitoring. Mechanics can register their service centers, specify their service types (such as towing, fuel delivery, battery jumpstart, tire replacement), and provide their service location with coordinates. Vehicle owners can submit emergency service requests by entering their vehicle details, location, and service requirements. The system includes a secure authentication system to ensure user authenticity and prevent misuse. A GPS-based location service is integrated to automatically find the nearest available mechanics based on real-time coordinates. Additionally, a feedback and rating system allows users to evaluate service quality after assistance. This project aims to digitalize and simplify the traditional method of finding roadside assistance, making it more efficient, reliable, and accessible for vehicle owners during emergencies

Contents

List of Figures

Chapter 1

Introduction

RoadMech is a web-based application developed using the Django framework to streamline and digitalize the process of providing on-road vehicle breakdown assistance. The platform serves three main user roles: Admin, Mechanic, and Customer. The Admin manages user accounts, approves mechanic registrations, and oversees service operations. Mechanics can register their service centers, specify the types of services offered (such as towing, fuel delivery, battery jumpstart, or tire replacement), provide their contact details, and set their service location. Customers can create accounts, choose their vehicle type, and request immediate roadside help by entering their address or using GPS-based location detection. The system then locates and displays nearby verified mechanics within a defined radius using geolocation and Haversine distance calculation. Once a mechanic is selected, customers can track their request, view service status, and submit feedback upon completion. The platform ensures secure authentication, real-time location mapping, and transparent communication between users, making it a reliable, fast, and user-friendly solution for roadside emergency assistance.

1.1 Background

1.1.1 Road Mech

The RoadMech project is a web-based system designed to modernize the process of on-road vehicle breakdown assistance. Traditionally, vehicle owners depend on nearby garages or acquaintances for help, leading to delays and uncertainty. RoadMech eliminates this problem by connecting vehicle owners directly with available mechanics through a secure and efficient digital platform. The system consists of three main components: the client, middleware, and database. The client side generates interactive web pages, handles user requests, and provides a smooth user experience, while the middleware functions as the web server, managing authentication, session handling, and coordination between the client and the database. The database stores all critical information, including user and mechanic profiles, service requests, feedback, and location data, with stored procedures ensuring data integrity and consistency in business operations. Additionally, JavaScript-based features enable real-time functionalities such as live location tracking, map integration, and instant service updates. Overall, RoadMech offers a reliable, fast, and organized solution for vehicle breakdown assistance, improving convenience and trust between customers and mechanics.

1.3 Motivation

Run on any operating system: - browser based application can be run on any computer which have fully functional browser. This cannot be adaptable for limited browser fictionalized devices(smart phone, PDAs)

No installation of client:-browser based applications do not need installation they only communicate through browser

1.4 Objective

- To digitalize the traditional roadside assistance process for vehicle breakdowns.
- To provide a fast, and reliable platform for connecting customers with nearby mechanics.
- To ensure location accuracy through GPS and map-based service requests.
- To enable customers to track service requests and mechanic responses in real time.
- To improve communication between customers and mechanics for quick and effective assistance

1.5 Contribution

The project aims to develop a browser-based system that eliminates the need for installing any additional client software on user devices. The Road Mech platform enables seamless communication between vehicle owners, mechanics, and the administrator through a web interface. JavaScript is used to manage client-side interactions, ensuring that user commands and responses are processed efficiently within the browser. The system also maintains session persistence by identifying returning users and securely storing essential client details. This approach allows users to access multiple features—such as requesting help, tracking mechanics, and managing profiles—within a single, user-friendly platform.

1.6 Report organization

The project report is divided into four sections. Section 2 describes literature survey. Section 3 describes the methodology used for implementing the project. Section 4 gives the results and discussions. Finally Section 5 gives the conclusion.

Chapter 2

Existing System

- Roadside assistance is mostly handled through manual calls or searching for nearby mechanics physically.
- There is no centralized platform to connect customers with available mechanics in real time.
- Customers face difficulty finding help quickly during vehicle breakdowns, especially in unfamiliar areas.
- Location tracking and accurate mechanic availability are not integrated in traditional methods.
- Communication between customers and mechanics is often delayed and unorganized.

Chapter 3

Methodology

3.1 Introduction

After the initial studies it is found that the agile model of software development is suitable and is the best method for the development of this system. Agile methodology mainly focused on the client satisfaction through continuous delivery. Also it sets a minimum number of requirements and turns them into a deliverable product. As this project has many individual requirements which can be delivered in parts and the user can gradually improve their work efficiency. Agile methodology has a family of methods of which scrum is selected for the development of this project. Scrum is a process framework that has been used to manage complex product development. It is not a process or technique for building products rather it is a framework within which various processes canbe employed. Also it is a suitable method to support the development process. It focuses on lean software development and is in building better software effectively and efficiently.Agile is one of the most widely used and recognized software development frameworks. The methodology those experts agreed upon was described as 'lightweight' and fast. Agile is also about being adaptive and continuous improvement, as much as it is about constant feedback and speed of delivery. Agile is a software development approach where a self-sufficient and cross-functional team works on making continuous deliveries through iterations and evolves throughout the process by gathering feedback from the end users.

1. The product owner (PO) : Who represents the stakeholder and the business.
2. The scrum master : Ensures the process followed, removes obstructions, and protects the developing system
3. Development team: Cross functional, self organizing team who actually do the actualanalysis,design implementation and testing process.

They work together in iterative time boxed durations called sprints. The first step is the creation of the product backlog by the PO. It's a to-do list of stuff to be done by the scrum team. Then the scrum team selects the top priority items and tries to finish them within the time box called a sprint. An easier way to remember all of this is to memorize the 3-3-5 framework. It means that a scrum project has 3 roles, 3 artifacts, and 5 events

These are:-

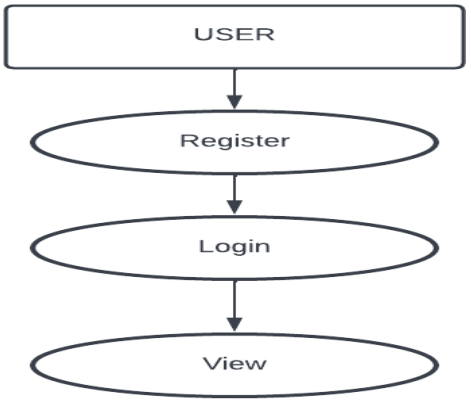
1. Roles : Product Owner, Scrum Master, and development team.
2. Artifacts : Product Backlog, Sprint Backlog and Product Increment.
3. Events : Sprint, Sprint planning, Daily Scrum, Sprint review and Sprint retrospective

The framework begins with a simple premise starting with what can be seen or known. After that the progress is tracked and tweaked as necessary. The three pillars of scrum aretransparency, inspection and adaptation. In scrum everyone has a role.

3.2 Work flow

Visual Studio is a cross platform code editor, and cloud based DevOps solutions, fully- featured IDE for building Android, iOS, Windows, web, and cloud apps in your favorite language.The Git is used as the version control system for this project. Version control is a system that records changes to a file or set of files over time so that a specific versions can be recalled later. Version control systems are a category of software tools that help a software team for managing changes to source code over time. Version control software keeps track of every modification to the code in a special kind of database. If a mistake is made, developers can turn back the clock and compare earlier versions of the code to help fix the mistake while minimizing disruption to all team members.

ROAD MECH is an application aims to simplify and modernize that process by connecting Customers and Laborers through a secure and user-friendly platform.. Admin can access the whole system and view the reports.



3.3 User story

USER STORY ID	AS A(TYPE OF USER)	I WANT TO	SO THAT I CAN
1	Admin	Login	Successfully access the admin dashboard with valid credential
2	Admin	Manage users	Verify or delete mechanic accounts
3	Admin	Manage requests	Monitor ongoing service requests and update their status
4	User	Registration	Register to get username and password
5	User	Login	Login successful with correct username and password
6	User	Request service	Request near by mechanics for on-road assistance
7	User	Track request	View mechanic location and service progress in real time
8	User	Give feedback	Provide service feedback after completion
9	User	Log out	Log out
10	Mechanic	Registration	Register and verify profile to receive service request
11	Mechanic	View request	View and accept or decline user service request
12	Mechanic	Update status	Update the job status
13	Mechanic	Log out	Log out

3.4 Product backlog

ID	PRIORITY	SIZE(HOURS)	SPRINT	STATUS	NAME
1	High	6	1	Planned	Registration
2	High	5	1	Planned	Login/Logout
3	High	10	2	Partially completed	Service request and tracking
4	Medium	8	2	Planned	Coding
5	Medium	6	3	Planned	Testing Data
6	Low	5	3	Planned	Output generation

3.5 Project plan

USER STORY ID	SPRINT	START DATE	END DATE	HOURS	STATUS
4,5,8,9	Sprint 1	01/07/2025	18/07/2025	11	Complete
1,2,3	Sprint 2	23/07/2025	30/07/2025	5	Complete
10,11	Sprint 3	01/08/2025	24/10/2025	14	Complete
12,13	Sprint 4	09/09/2025	26/09/2025	12	Complete
6,7	Sprint 5	03/10/2025	24/10/2025	14	Complete

3.6 Sprint backlog

Sprint 1:

Backlog item	Status and completion date	Original estimate in hours	Day 1 01/07	Day 2 02/07	Day 3 04/07	Day 4 08/07	Day 5 11/07	Day 6 15/07	Day 7 16/07	Day 8 18/07
Form Design	02/07	2	1	1	0	0	0	0	0	0
Table Design	04/07	2	0	0	2	0	0	0	0	0
Coding	08/07	5	0	0	0	1	2	1	0	1
Testing & validation	08/07	2	0	0	0	0	0	0	1	1
Total		11	1	1	2	1	2	1	1	2

Sprint 2:

Backlog item	Status and completion date	Original estimate in hours	Day 1 23/07	Day 2 25/07	Day 3 29/07	Day 4 30/07
Coding	29/07	2	1	0	1	0
Testing & validation	30/07	3	0	2	0	1
Total		5	1	2	1	1

Sprint 3:

Backlog item	Status and completion date	Original estimate in hours	Day 1 01/08	Day 2 05/08	Day 3 06/08	Day 4 08/08	Day 5 12/08	Day 6 13/08	Day 7 19/08	Day 8 20/08	Day 9 22/08	Day 10 26/08	Day 11 27/08
Form Design	01/08	2	2	0	0	0	0	0	0	0	0	0	0
Table Design	06/08	2	0	1	1	0	0	0	0	0	0	0	0
Coding	26/08	8	0	0	0	2	1	1	1	1	1	1	0
Testing & validation	27/08	2	0	0	0	0	0	0	0	0	1	0	1
Total		14	2	1	1	2	1	1	1	1	2	1	1

Sprint 4:

Backlog item	Status and completion date	Original estimate in hours	Day 1 09/09	Day 2 10/09	Day 3 12/09	Day 4 16/09	Day 5 17/09	Day 6 19/09	Day 7 23/09	Day 8 24/09	Day 9 26/09
Form Design	10/09	2	1	1	0	0	0	0	0	0	0
Coding	26/09	8	0	0	2	1	1	2	1	0	1
Testing & validation	26/09	2	0	0	0	0	0	0	0	1	1
Total		12	1	1	2	1	1	2	1	1	2

Sprint 5:

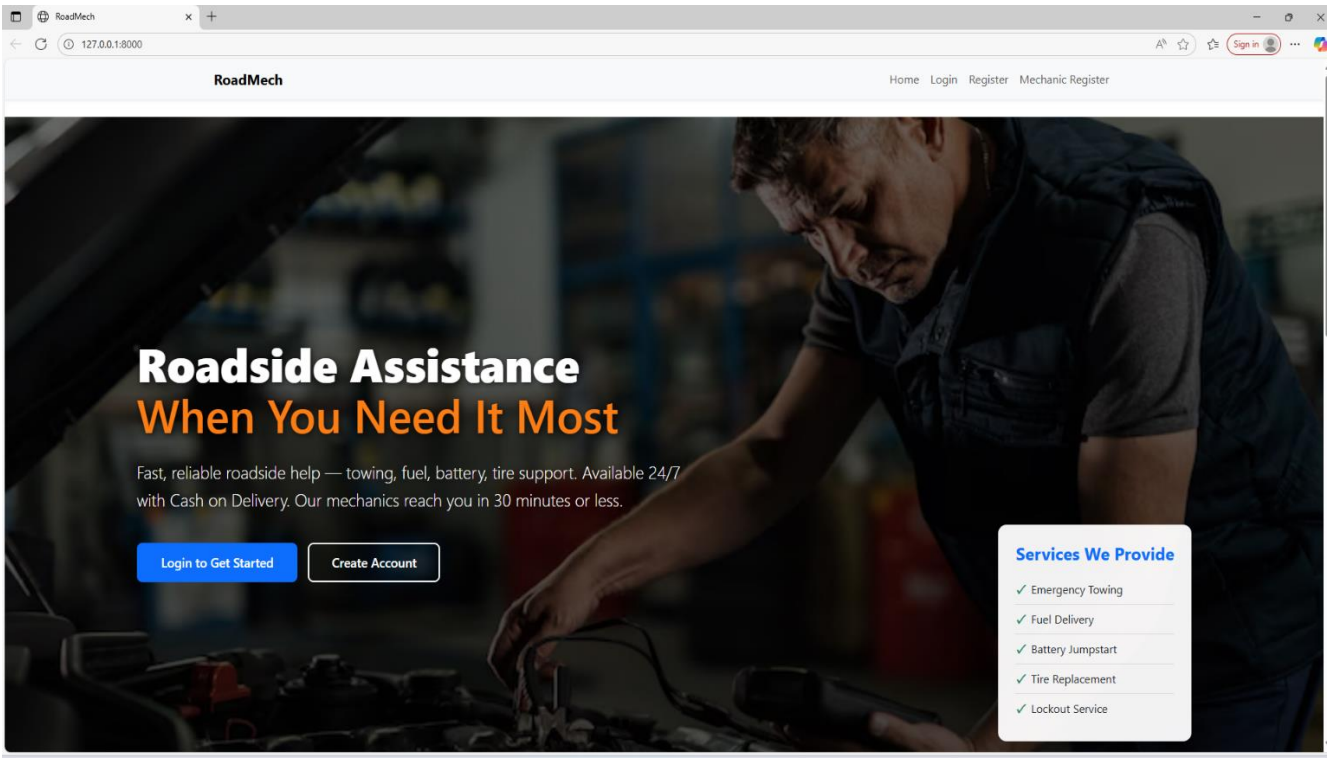
Backlog item	Status and completion date	Original estimate in hours	Day 1 03/10	Day 2 07/10	Day 3 08/10	Day 4 10/10	Day 5 14/10	Day 6 15/10	Day 7 17/10	Day 8 121/10	Day 9 22/10	Day 10 24/10
Form Design	03/10	2	2	0	0	0	0	0	0	0	0	0
Table Design	08/10	2	0	1	1	0	0	0	0	0	0	0
Coding	24/10	8	0	0	0	2	1	1	2	1	0	1
Testing & validation	24/10	2	0	0	0	0	0	0	0	0	1	1
Total		14	2	1	1	2	1	1	2	1	1	2

Chapter 4

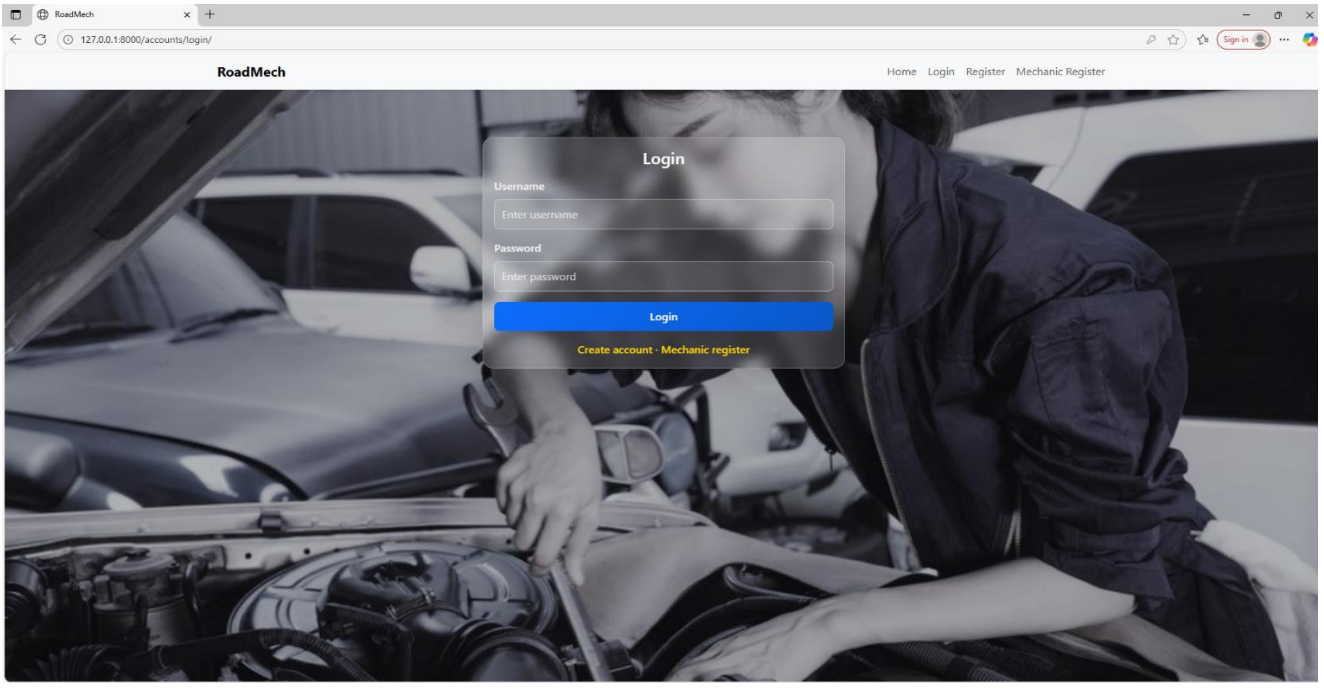
Design

4.1 Result

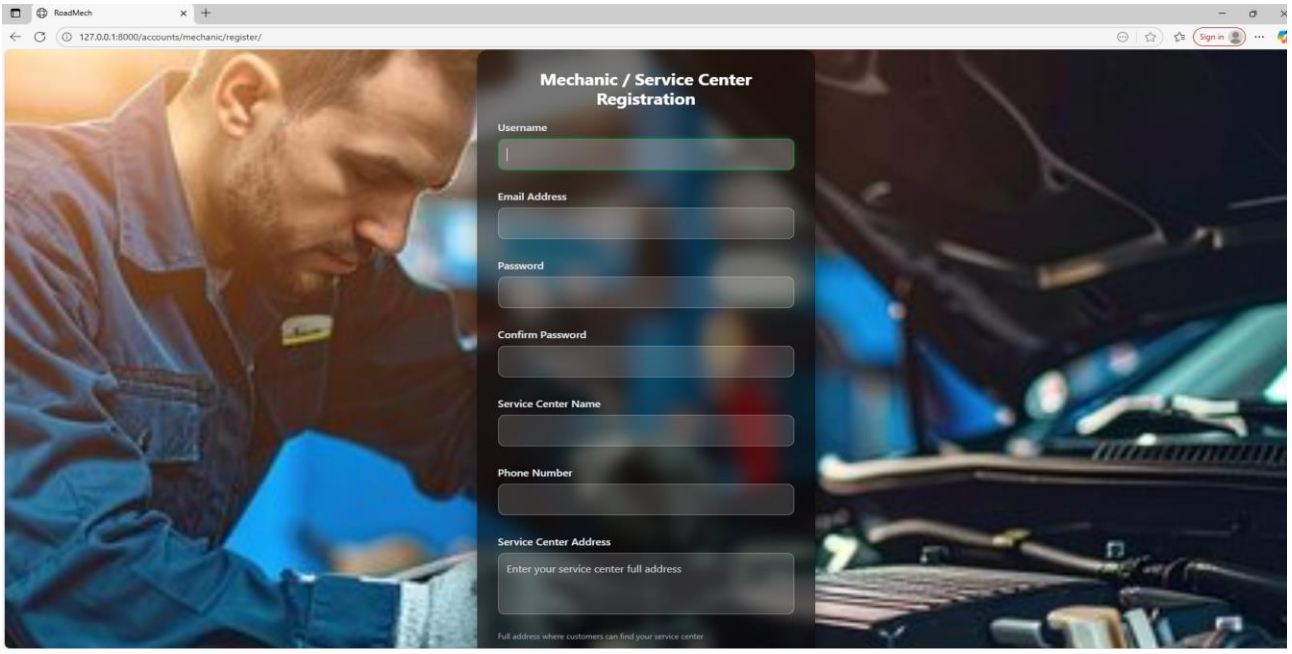
Index

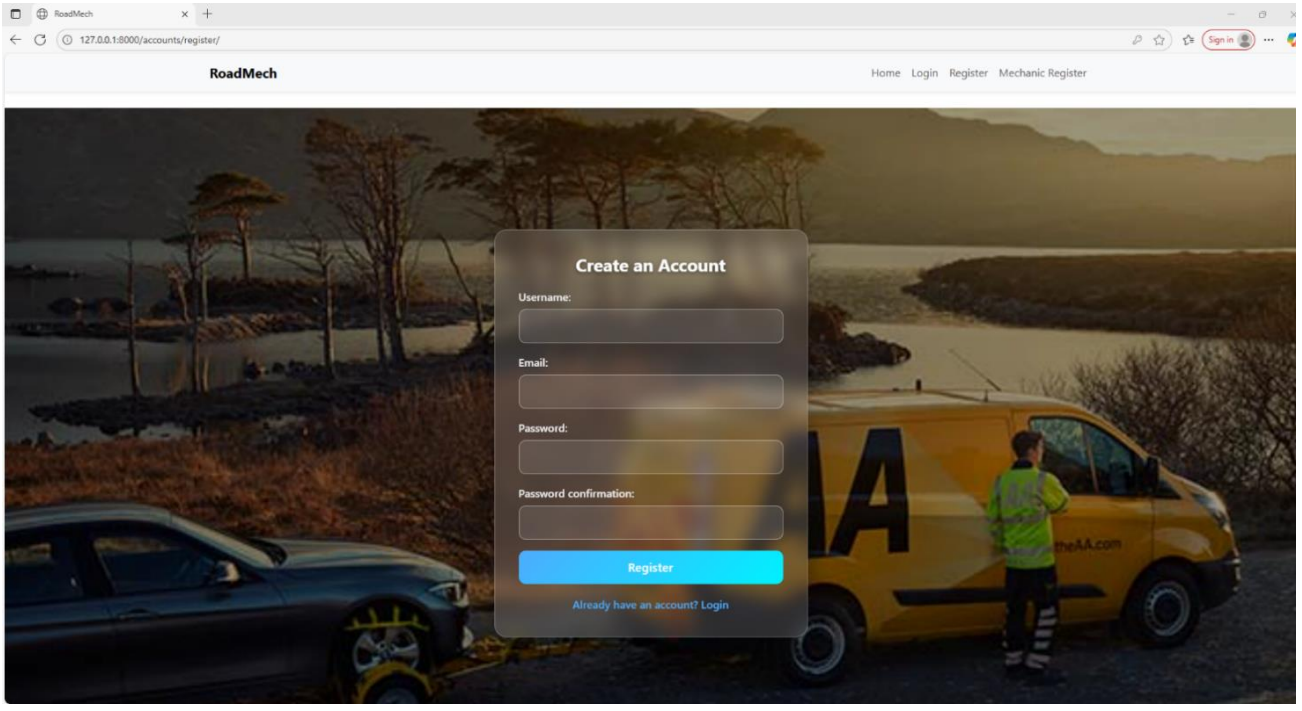


Login

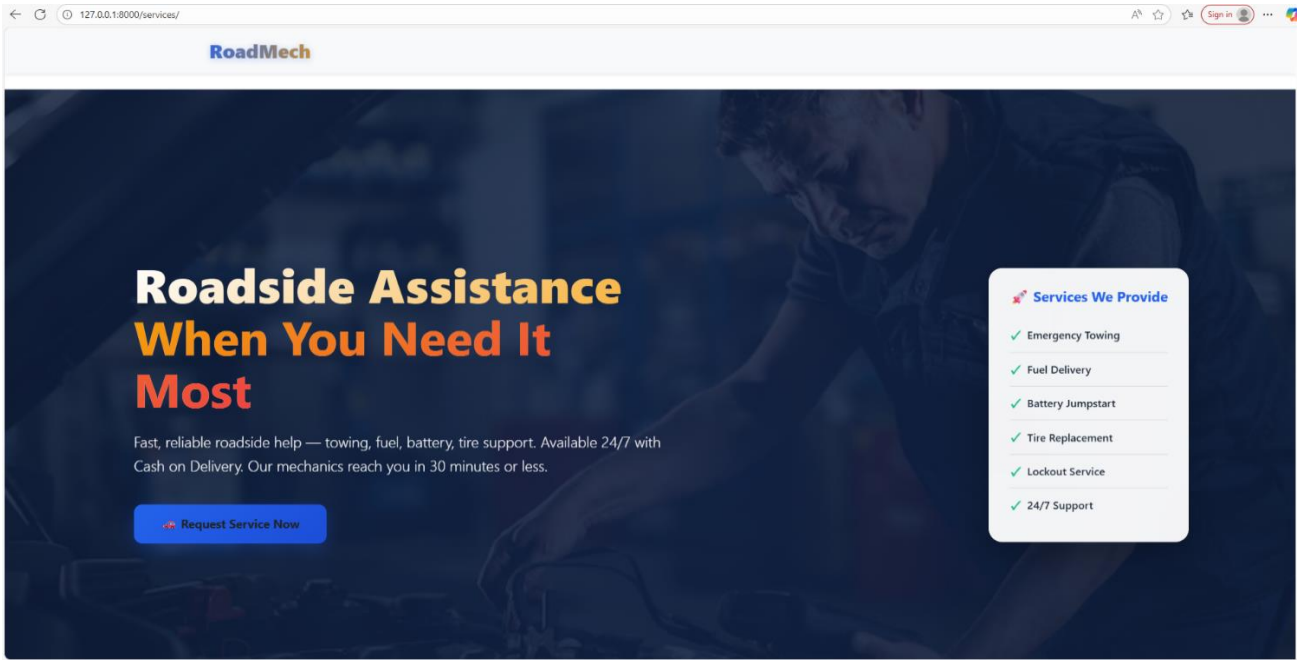


Registation

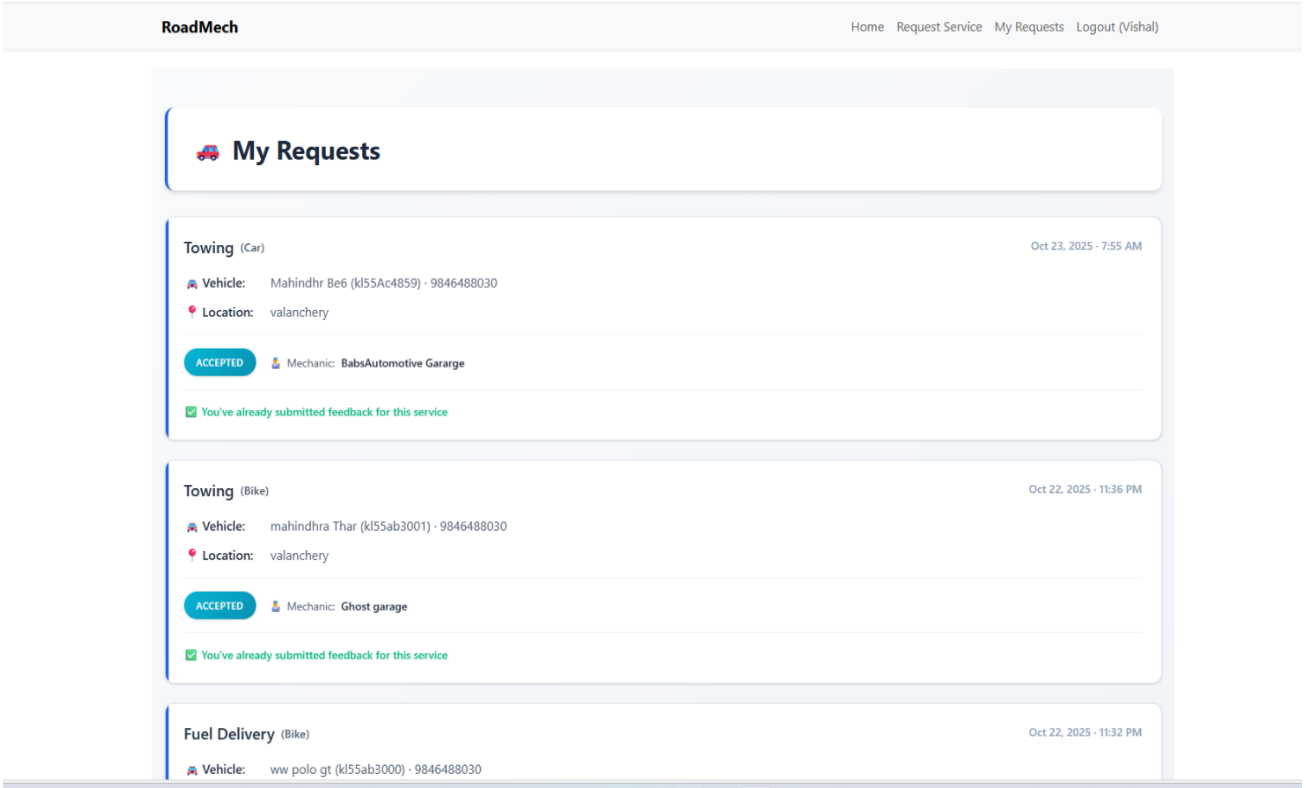




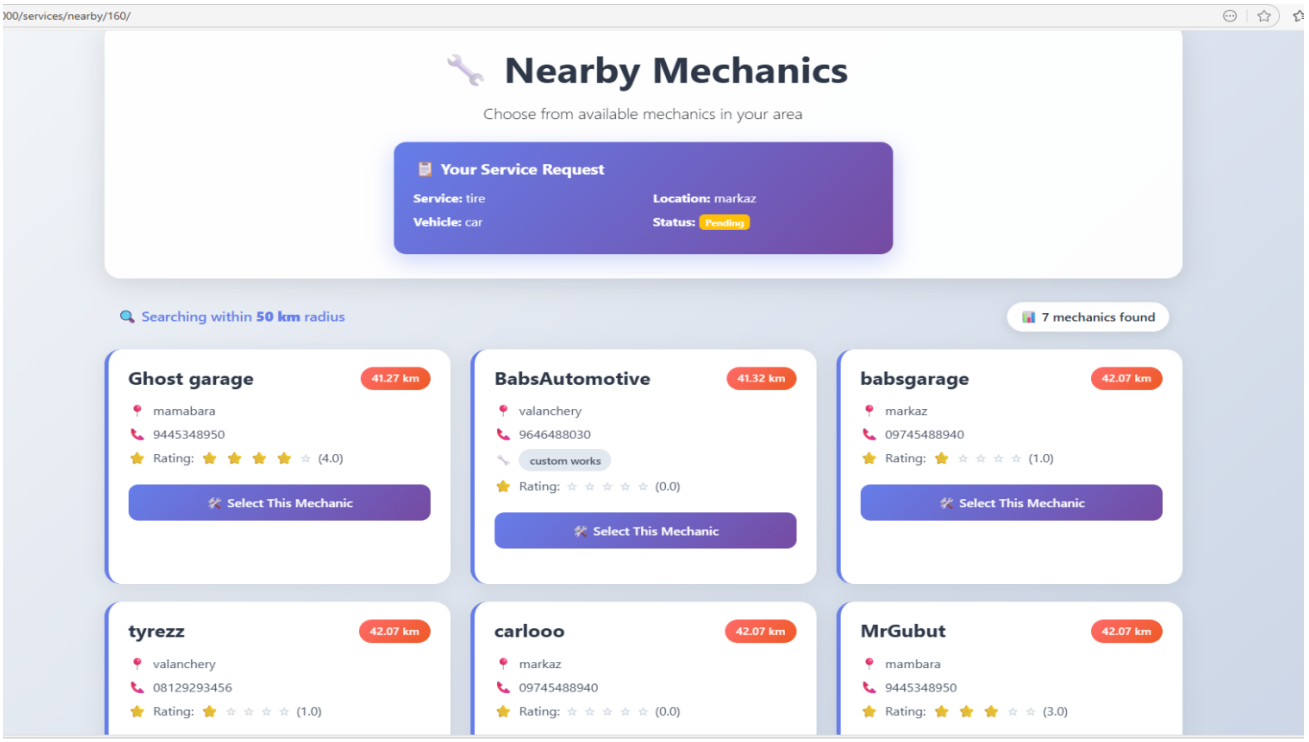
User Home



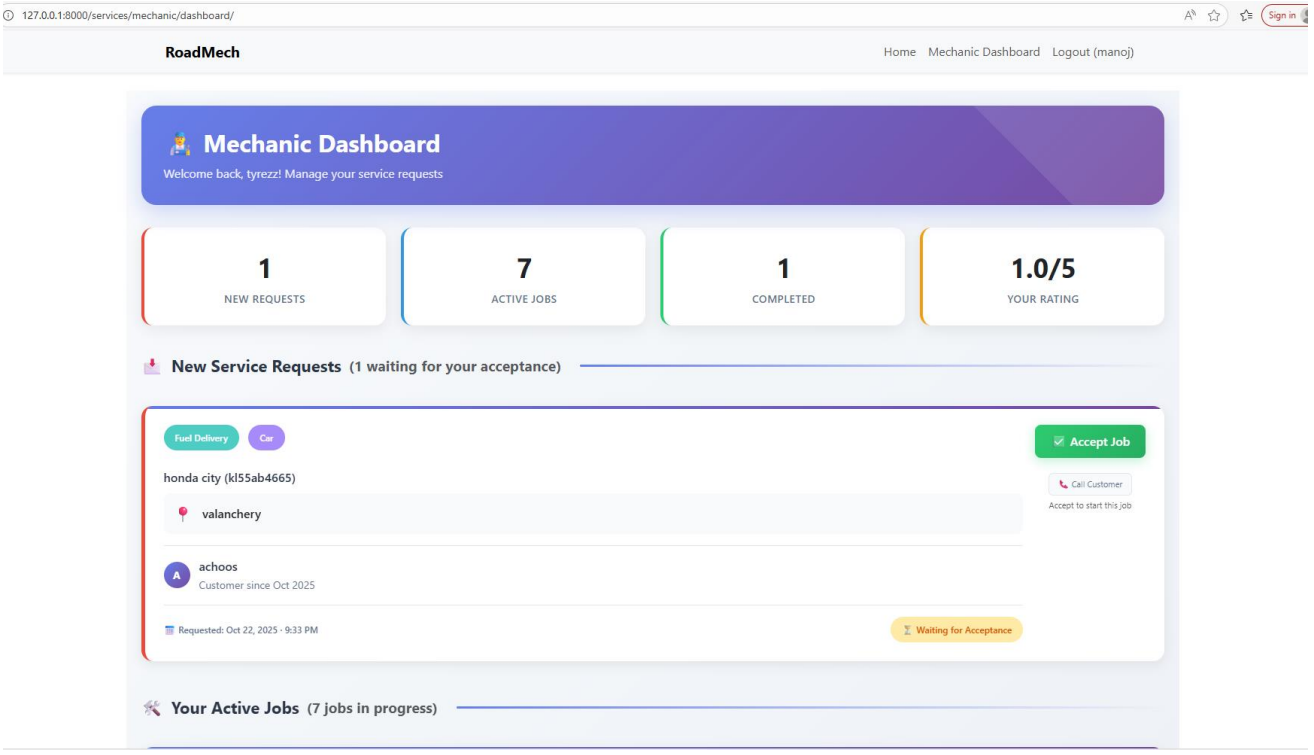
User requests



User Nearby Mechanics Search



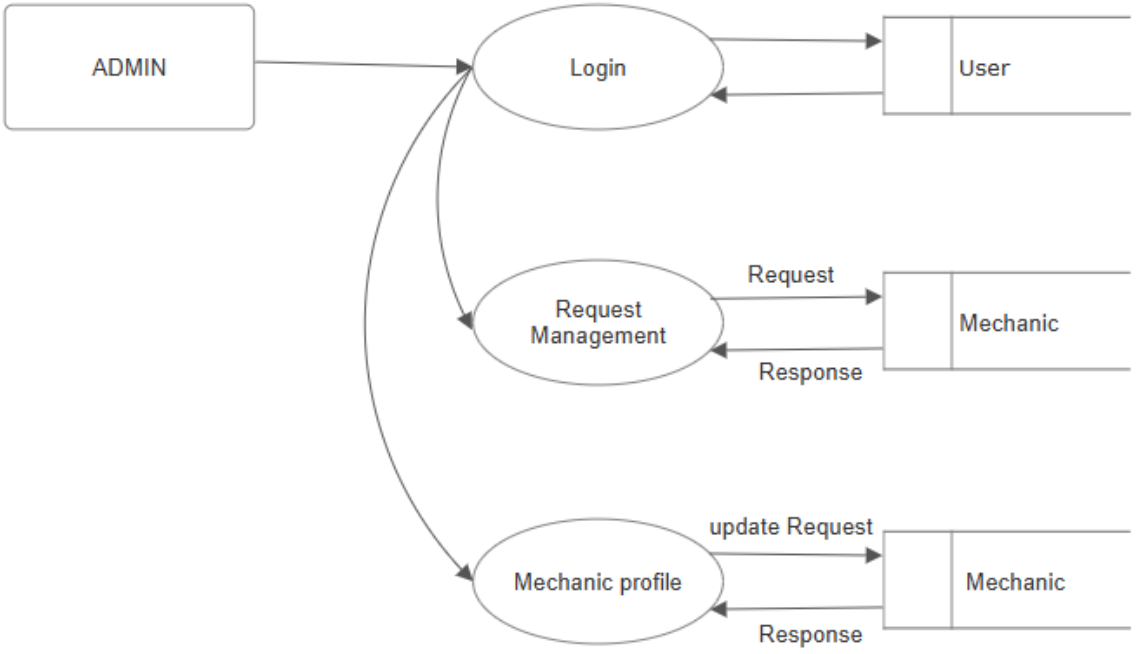
Mechanic Home



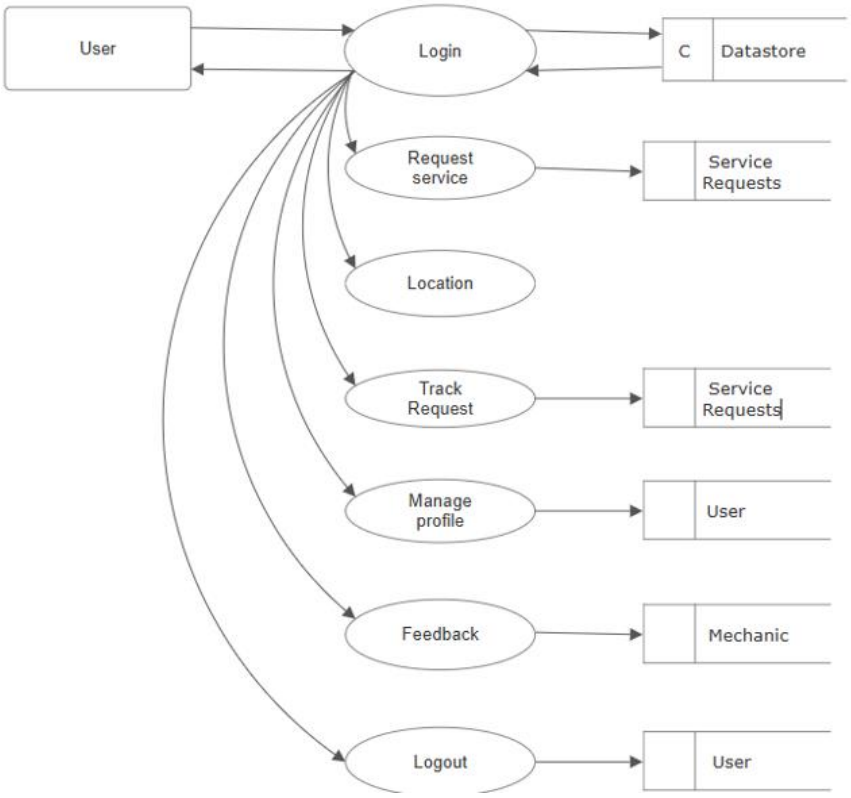
4.2 Data Flow Diagram

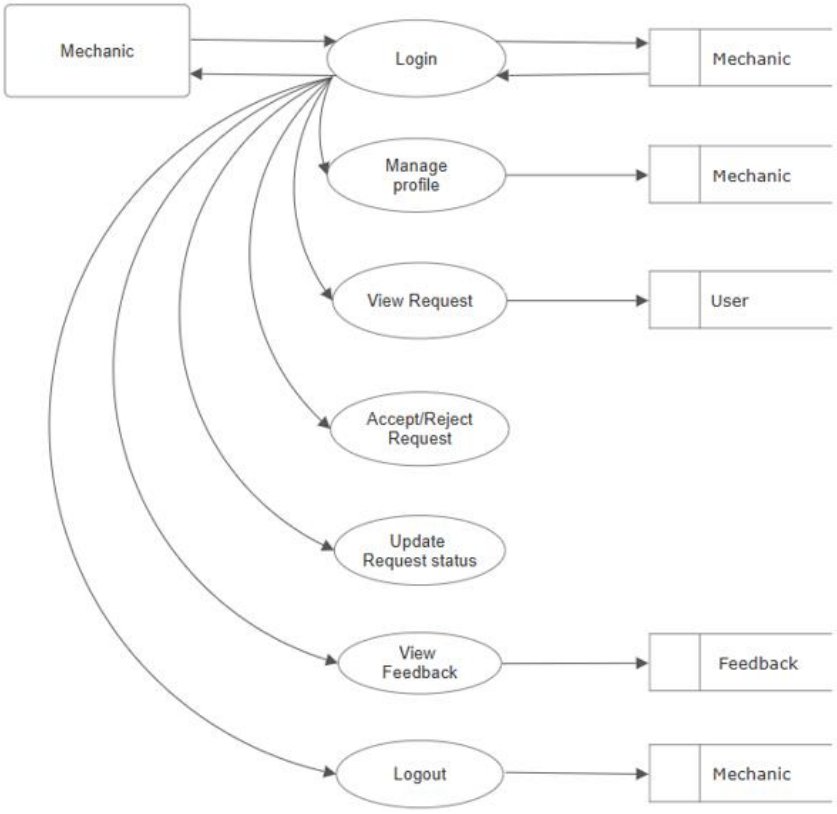
Level 0 :

Level 1 :

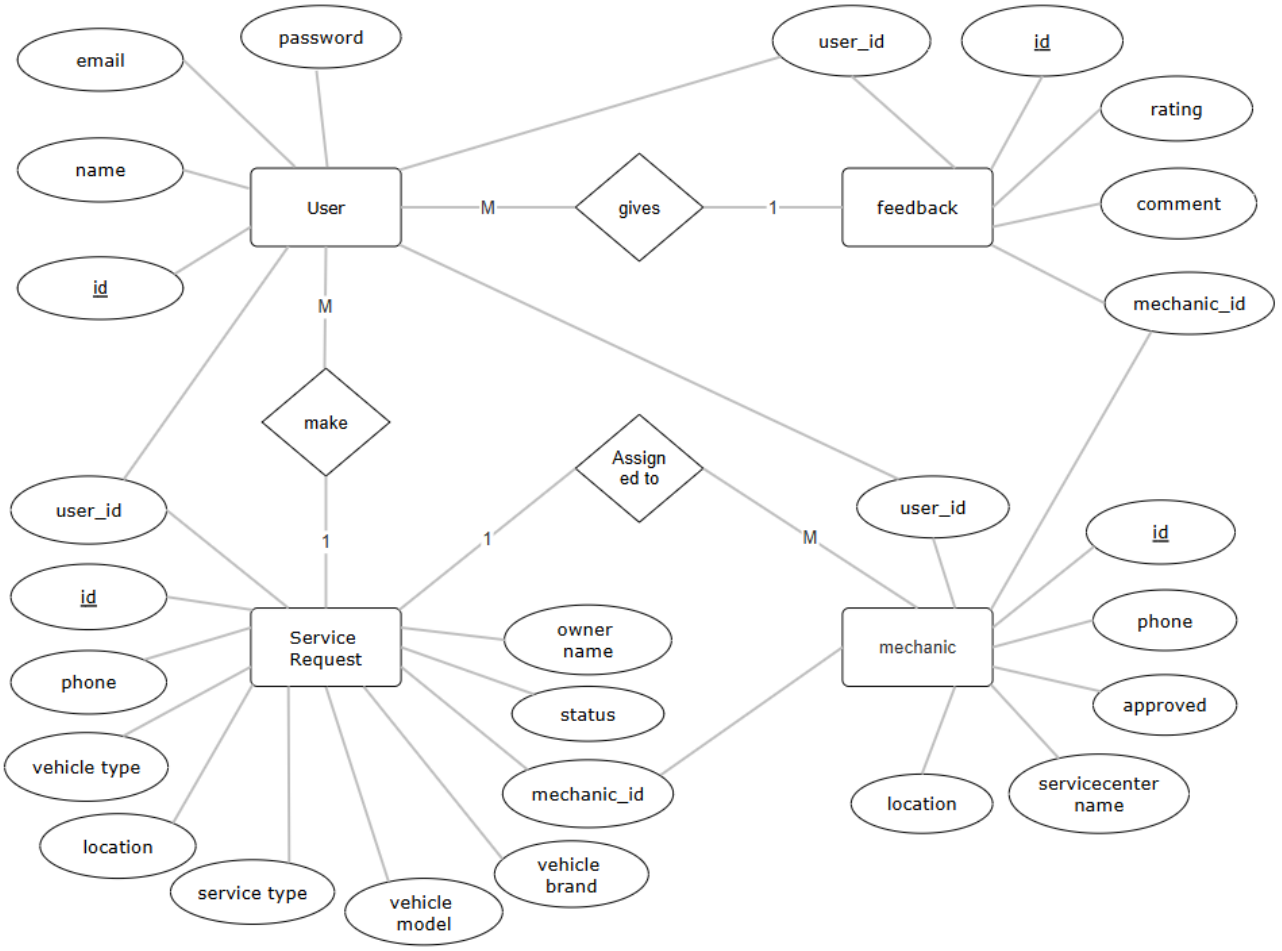


Level 2 :


User
Login
C
Datastore
Request service
Service Requests
Location
Track Request
Service Requests
Manage profile
User
Feedback
Mechanic
Logout
User


Mechanic
Login
Mechanic
Manage profile
Mechanic
View Request
User
Accept/Reject Request
Update Request status
View Feedback
Feedback
Logout
Mechanic

4.3 ER Diagram



4.4 Table

Mechanic

Field	Type	Description
id	Integer (Primary Key)	Auto_increment ID
user	OneToOneField(User)	Link to Django user
service_center_name	CharField(255)	Name of mechanic's shop
phone	CharField(20)	Contact number
location	CharField(255), nullable	Mechanich's address
latitude	FloatField	GPS lattitude
longitude	FloatField	GPS longitude
approved	BooleanField	Admin approval status
Created_at	DateTimeField	Record creation time

Feedback

Field	Type	Description
id	Integer(primary key)	Auto-increment ID
user	ForeignKey(User)	Feedback given by customer
mechanic	ForeignKey(Mechanic Profile)	Mechanic receiving feedback
rating	PositiveIntegerField	Rating(1-5)
comment	TextField	Feedback comment
Created_at	Date Time Field	Time feedback submitted

ServiceRequest

Field	Type	Description
id	Integer (Primary Key)	Auto_increment ID
user	ForeignKey(User)	Request made by user
mechanic	ForeignKey(MechanicProfile, null=True)	Assigned mechanic
vehicle_type	CharField(10)	Car/Bike
Service_type	CharField(20)	Type of service requested
Vehicle_brand	CharField(100)	Brand name
Vehicle_model	CharField(50)	Model name
Vehicle_year	Positive	Manufacturing year
Vehicle_number	CharField(20)	Vehicle registration number
Owner_name	CharField(100)	Owner's name
Phone_number	CharField(15)	Owner's contact
location	CharField(255)	Address/location of breakdown
latitude	DecimalField(9,6)	GPS latitude
longitude	DecimalField(9,6)	GPS longitude
status	CharField(20)	Pending/Accepted/Completed
Created_at	Date Time Field	Timestamp of request

Chapter 5

CONCLUSION

Although this project is only to construct a JavaScript based client for the software development and implementation using Itec Cloud architecture, the success and adaptation of this Architecture is crucial to the usage of this JavaScript based client.One reason for the slow adaptability of this architecture was that the client has to be Installed on each user's computer. Now the client is browser based, it will surely pickup.Development is faster than developing using existing Ajax based development tools.Ex ecution is also faster since the business code is running inside the database engine. Sql com mands can be directly written inside PL/SQL while this facility is not available in any language running outside a database. More over various PL/SQL utilities are also provided with Itec Cloud products. It saves the time for writing login and menu programs.Further the code size is much smaller than codes written in PHP and other languages Used to develop web based applications. Disk access time is greater in the order of thousand When compared execution time. Hence there is much saving in time in loading a program from disk. The command structure is very useful for writing reports. Definition of report layout, header and tail enables the client to change to next page without any command from the backend. Commands are also available for defining total columns which will enable the client to calculate totals.

5.1 References

[1] Adrian Holovaty and Jacob Kaplan-Moss. The Definitive Guide to Django: Web Development Done Right. Apress, Second Edition, 2009.

[2] William S. Vincent. Django for Beginners: Build websites with Python and Django. 2023 Edition.

[3] Allen B. Downey. Think Python: How to Think Like a Computer Scientist. O'Reilly Media, 2015.

[4] Mark Lutz. Learning Python. O'Reilly Media, Fifth Edition, 2013.

[5] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. Database System Concepts. McGraw-Hill, Seventh Edition, 2020.

[6] Elmasri, Ramez and Navathe, Shamkant. Fundamentals of Database Systems. Pearson, Seventh Edition, 2017.

[7] Journal of Web Engineering, Vol. 16(5), "Modern Web Application Architectures using Python and Django Frameworks", 2020.

[8] Journal of Computer Applications, Vol. 8(2): "Role-Based Authentication in Web Applications using Django ORM", 2021.

[9] Django Software Foundation. Django Official Documentation. Retrieved from: https://docs.djangoproject.com/

[10] Python Software Foundation. Python 3 Documentation. Retrieved from: https://docs.python.org/3/

[11] W3Schools. Django Tutorial. Retrieved from: https://www.w3schools.com/django/

[12] MDN Web Docs. HTML, CSS, and JavaScript Reference. Retrieved from: https://developer.mozilla.org/

[13] TutorialsPoint. Django Framework Tutorial. Retrieved from: https://www.tutorialspoint.com/django/

[14] GeeksforGeeks. Django Tutorials and Projects. Retrieved from: https://www.geeksforgeeks.org/django-tutorial/

[15] DigitalOcean. How to Build and Deploy Django Applications. Retrieved from: https://www.digitalocean.com/community/tutorials

[16] GitHub. Open-source Django Projects Repository. Retrieved from: https://github.com/django

[17] RealPython. Practical Python and Django Tutorials. Retrieved from: https://realpython.com/

Chapter 6

Appendix

6.1 Source Code

Services

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from django.db.models import Avg
from .forms import ServiceRequestForm, FeedbackForm
from .models import ServiceRequest, Feedback
from accounts.models import MechanicProfile
import math

def home(request):
    return render(request, 'home.html')

@login_required
def choose_vehicle(request):
    return render(request, 'services/choose_vehicle.html')

def haversine_km(lat1, lon1, lat2, lon2):
    """Return distance in kilometers between two lat/lon points using Haversine."""
    if None in (lat1, lon1, lat2, lon2):
        return None
    R = 6371.0  # Earth's radius in kilometers
    lat1_r, lon1_r = math.radians(lat1), math.radians(lon1)
    lat2_r, lon2_r = math.radians(lat2), math.radians(lon2)
    dlat = lat2_r - lat1_r
    dlon = lon2_r - lon1_r
    a = math.sin(dlat/2)**2 + math.cos(lat1_r) * math.cos(lat2_r) * math.sin(dlon/2)**2
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    return R * c

@login_required
def request_service(request):
    """
    GET: show form
    POST: save ServiceRequest (including lat/lon) then redirect to nearby_mechanics view
    """
    vehicle_type = request.GET.get("vehicle")  # from choose_vehicle
```

```
 if request.method == "POST":
        form = ServiceRequestForm(request.POST)
        if form.is_valid():
            sr = form.save(commit=False)
            sr.user = request.user
            if vehicle_type:
                sr.vehicle_type = vehicle_type


            lat = request.POST.get("latitude") or request.POST.get("lat")
            lon = request.POST.get("longitude") or request.POST.get("lon")
            try:
                sr.latitude = float(lat) if lat not in (None, "") else None
                sr.longitude = float(lon) if lon not in (None, "") else None
            except (ValueError, TypeError):
                sr.latitude = None
                sr.longitude = None

            sr.save()

            return redirect('nearby_mechanics', request_id=sr.id)
        else:

            messages.error(request, "Please fix the errors in the form.")
    else:
        initial = {}
        if vehicle_type:
            initial["vehicle_type"] = vehicle_type
        form = ServiceRequestForm(initial=initial)

    return render(request, "services/request_service.html", {"form": form, "vehicle_type":
vehicle_type})

@login_required
def nearby_mechanics(request, request_id):
    """Show nearby mechanics for a saved ServiceRequest"""
    sr = get_object_or_404(ServiceRequest, id=request_id, user=request.user)

    if sr.latitude is None or sr.longitude is None:
        messages.error(request, "Location not set on your request. Please allow geolocation or enter a
location.")
        return redirect('request_service')


    mechanics_qs =
MechanicProfile.objects.filter(approved=True).exclude(latitude__isnull=True).exclude(longitude__i
snull=True)
    nearby = []
    for m in mechanics_qs:
        if m.latitude is None or m.longitude is None:
            continue
```

```
 dist = haversine_km(sr.latitude, sr.longitude, m.latitude, m.longitude)
        if dist is None:
            continue
        if dist <= 50:  # 50 km (change as you want)
            nearby.append((m, dist))

    nearby.sort(key=lambda x: x[1])
    return render(request, "services/nearby_mechanics.html", {"sr": sr, "nearby": nearby})

@login_required
def assign_mechanic(request, request_id, mechanic_id):
    sr = get_object_or_404(ServiceRequest, pk=request_id, user=request.user)
    mech = get_object_or_404(MechanicProfile, pk=mechanic_id, approved=True)
    sr.mechanic = mech
    sr.status = "Accepted"
    sr.save()
    messages.success(request, "Mechanic assigned to your request.")
    return redirect("user_dashboard")

@login_required
def user_dashboard(request):
    reqs = ServiceRequest.objects.filter(user=request.user).order_by('-created_at')
    return render(request, 'services/user_dashboard.html', {'requests': reqs})

@login_required
def accept_request(request, pk):
    try:
        mp = request.user.mechanicprofile
    except Exception:
        messages.error(request, "Only registered mechanics can accept requests.")
        return redirect('home')

    sr = get_object_or_404(ServiceRequest, pk=pk)
    if sr.status != 'Pending':
        messages.warning(request, "This request is not available to accept.")
        return redirect('mechanic_dashboard')
    sr.mechanic = mp
    sr.status = 'Accepted'
    sr.save()
    messages.success(request, "You accepted the request.")
    return redirect('mechanic_dashboard')

def service_success(request):
    return render(request, "services/service_success.html")

@login_required
def give_feedback(request, mechanic_id):
    mechanic = get_object_or_404(MechanicProfile, id=mechanic_id)
```

```
  if request.method == "POST":
        form = FeedbackForm(request.POST)
        if form.is_valid():
            feedback, created = Feedback.objects.update_or_create(
                user=request.user,
                mechanic=mechanic,
                defaults={'rating': form.cleaned_data['rating'], 'comment':
form.cleaned_data.get('comment', '')}
            )
            messages.success(request, "Thanks for your feedback.")
            return redirect('mechanic_detail', mechanic_id=mechanic.id)
    else:
        form = FeedbackForm()
    return render(request, 'services/give_feedback.html', {'form': form, 'mechanic': mechanic})

def mechanic_detail(request, mechanic_id):
    mechanic = get_object_or_404(MechanicProfile, id=mechanic_id)
    feedbacks = mechanic.feedbacks.all()
    avg_rating = feedbacks.aggregate(Avg("rating"))["rating__avg"] or 0
    return render(request, 'services/mechanic_detail.html', {
        'mechanic': mechanic,
        'feedbacks': feedbacks,
        'avg_rating': round(avg_rating, 1)
    })

@login_required
def search_mechanics(request):
    lat = request.GET.get("lat")
    lon = request.GET.get("lon")
    radius_km = float(request.GET.get("radius", 10))
    mechanics_list = []

    if lat and lon:
        try:
            lat_f = float(lat); lon_f = float(lon)
        except ValueError:
            lat_f = lon_f = None

        if lat_f is not None:
            qs =
MechanicProfile.objects.filter(approved=True).exclude(latitude__isnull=True).exclude(longitude__i
snull=True)
            for mech in qs:
                if mech.latitude is None or mech.longitude is None: continue
                distance = haversine_km(lat_f, lon_f, mech.latitude, mech.longitude)
                if distance <= radius_km:
                    mechanics_list.append((mech, distance))
            mechanics_list.sort(key=lambda x: x[1])
```

```
    return render(request, "services/search_mechanics.html", {
        "query": f"{lat},{lon}" if lat and lon else "",
        "radius": radius_km,
        "mechanics": mechanics_list,
    })
```

Accounts

```
from django.shortcuts import render, redirect
from django.contrib import messages
from .forms import UserRegistrationForm, MechanicRegistrationForm
from django.contrib.auth import login
from django.contrib.auth.decorators import login_required
from services.models import ServiceRequest
from .models import MechanicProfile
from utils.geocode import geocode_address

def register(request):
    if request.method == 'POST':
        form = UserRegistrationForm(request.POST)
        if form.is_valid():
            user = form.save()
            messages.success(request, "Registration successful. You can now log in.")
            return redirect('login')
    else:
        form = UserRegistrationForm()
    return render(request, 'accounts/register.html', {'form': form})

# In your views.py
def mechanic_register(request):
    if request.method == 'POST':
        form = MechanicRegistrationForm(request.POST)
        if form.is_valid():
            user = form.save()

            # Create MechanicProfile with all required fields
            mp = MechanicProfile.objects.create(
                user=user,
                service_center_name=form.cleaned_data['service_center_name'],
                phone=form.cleaned_data['phone'],
                location=form.cleaned_data['location'],
                latitude=form.cleaned_data.get('latitude'),
                longitude=form.cleaned_data.get('longitude'),
                approved=False  # Mechanic needs to be approved by admin
            )
```

```
  # Login the user or redirect to login page
            # login(request, user)
            messages.success(request, 'Mechanic registration successful! Waiting for approval.')
            return redirect('login')
    else:
        form = MechanicRegistrationForm()

    return render(request, 'accounts/mechanic_register.html', {'form': form})

@login_required
def mechanic_dashboard(request):
    # only mechanics should access: ensure they have a MechanicProfile
    try:
        mp = request.user.mechanicprofile
    except Exception:
        messages.error(request, "You must register as a mechanic to view this page.")
        return redirect('home')

    pending_requests = ServiceRequest.objects.filter(status='Pending')
    my_accepted = ServiceRequest.objects.filter(mechanic=mp)
    context = {'pending_requests': pending_requests, 'my_accepted': my_accepted}
    return render(request, 'services/mechanic_dashboard.html', context)

@login_required
def redirect_after_login(request):
    # send mechanic to mechanic dashboard
    if hasattr(request.user, "mechanicprofile"):
        return redirect("mechanic_dashboard")
    # normal users go to home
    return redirect("home")
```