

PROJECT REPORT

Name: Askar E Z, Vibhish R, Vigneshrajan S

Email: vigneshrajan.s@prodapt.com, zainulaabideen.z@prodapt.com,
vibhish.r@prodapt.com,

WHEEL'S,UP – Home Automobile Service

Abstract

- Our Project approves presenting car and Bike service at the door-step. If consumer obtained any hassle with his automobile based totally on **assurance card**, they are offering services.
- The actual energy of this mission lies now not in direct promoting of products, however in the advent of **tighter relationships** with clients and handing over of an excessive stage of carrier and support, which in flip improves employer income and its goodwill.
- A provider organization is a commercial enterprise entity that takes care of servicing a purchaser instrument in the after-income domain.
- As the variety of clients and dimension of operations increases, the organization divides the geographical place into provider areas and department locations, to enable Engineers to be greater responsive to the customer-needs.
- Our Aim is to ease the **customer problem** and have a **friendly response** to their query at their convenience.

Problem Statement

- As you know it is tedious and annoying to have a breakdown or repair in our car or bike when you are reaching somewhere urgent. In general, we have to go to a mechanic shop or call him and he may be not available at the time we need him.
- Today we have everything at the disposal of our hand and is even available at our **door-step** from clothes, meat and even pets which we generally call **E-Commerce** but why not the service we required for our vehicles at our door-step.....?
- Above problems can be seen as from point of view from a customer but what from the service provider it is not easy to just provide service right so what we can do to ease their side of inconveniences and issues.
- From a service provider view other than providing a service he has to maintain the logs and record of services and sometimes in case of any issues he has to cross-check the records from the pile which is not easy and lot more tedious work to go.
- **What we know from this:**
 - From first point, it is clear that a handy service for automobile is not available for customer at their convenience.
 - Then even though we can call a repair service there isn't a proper flow all the time and it is not guarantee that we can reach to them all the time.
 - As from service provider side, fetching and storing customer details from the pile of record and keeping track of work is not easy and tedious.
- Next, we will see the possible solution to problems mentioned above.

Proposed Solution

- What we propose is to give customer the needed service at their **Doorstep** and their convenience.
- For which we provide a **user-friendly interface** with the features intended with a database which is convenient for service-provider let's see the actual process and logic involved behind them.

Business Logic:

- We have divided the logic into 3 modules:
 - User Registration
 - Services Needed
 - Booking Module
- **User Registration:** As there is need to store the data for users so there is necessity for them to register and log-in to home page in order to access the further service and assistance. This also serves as a **security purpose** for user's confidentiality and ease in **storing data** which can be convenient for service provider.
- **Services Needed:** Then comes the main part which will be the service from which customer can select their desired service depending on their need and also if there is any query which is not addressed in our service then the customer can also use our **Live-chat** feature to address their query which we will respond immediately to help at their convenience.
- **Booking Module:** Then comes the final step which is booking the service the customer opted which is done through getting their address, mobile no, vehicle type, manufacturer and model. Then after booking they can see their booking preview in order to know what is happening.

Data Required:

- This process is done during booking during which we get user's data like vehicle type, vehicle manufacturer, vehicle model and vehicle fuel.
- Which helps in identifying the data of the user and is useful while providing service.

Decision made based on the Output:

- What we can derive from this is that data retrieved can be used for future analysis of the problem with the same user or any other user with the same problem.

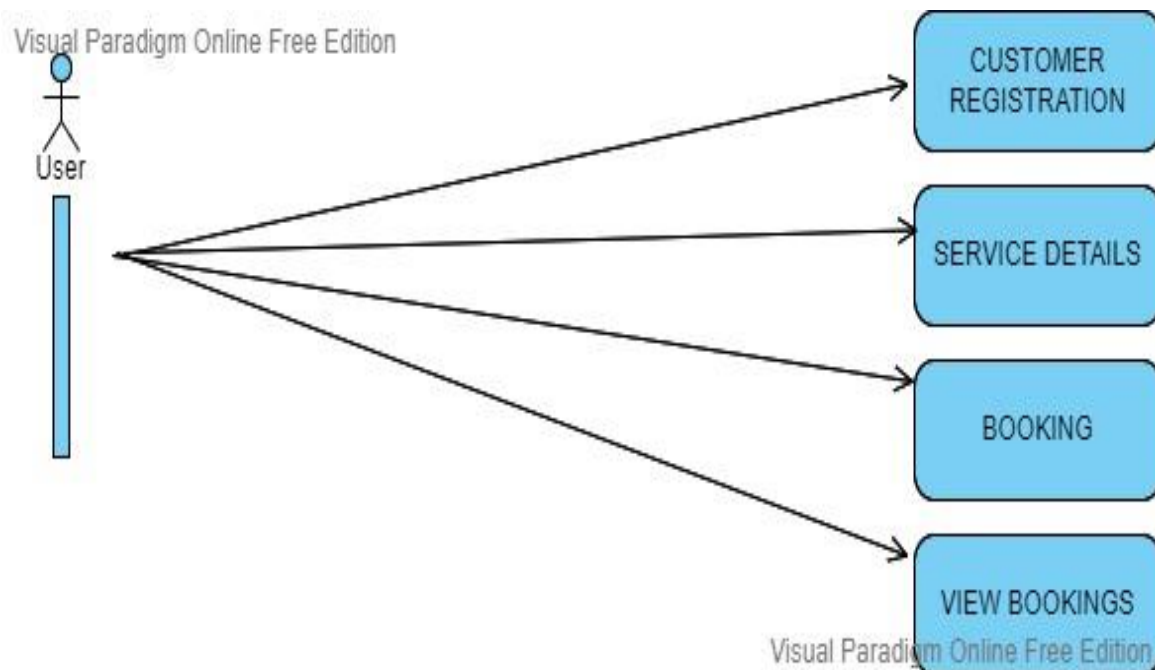
- The major advantage of this is that it is convenient for user to access the service anytime even during the emergency time the service can be accessed easily.

Framework and Database used:

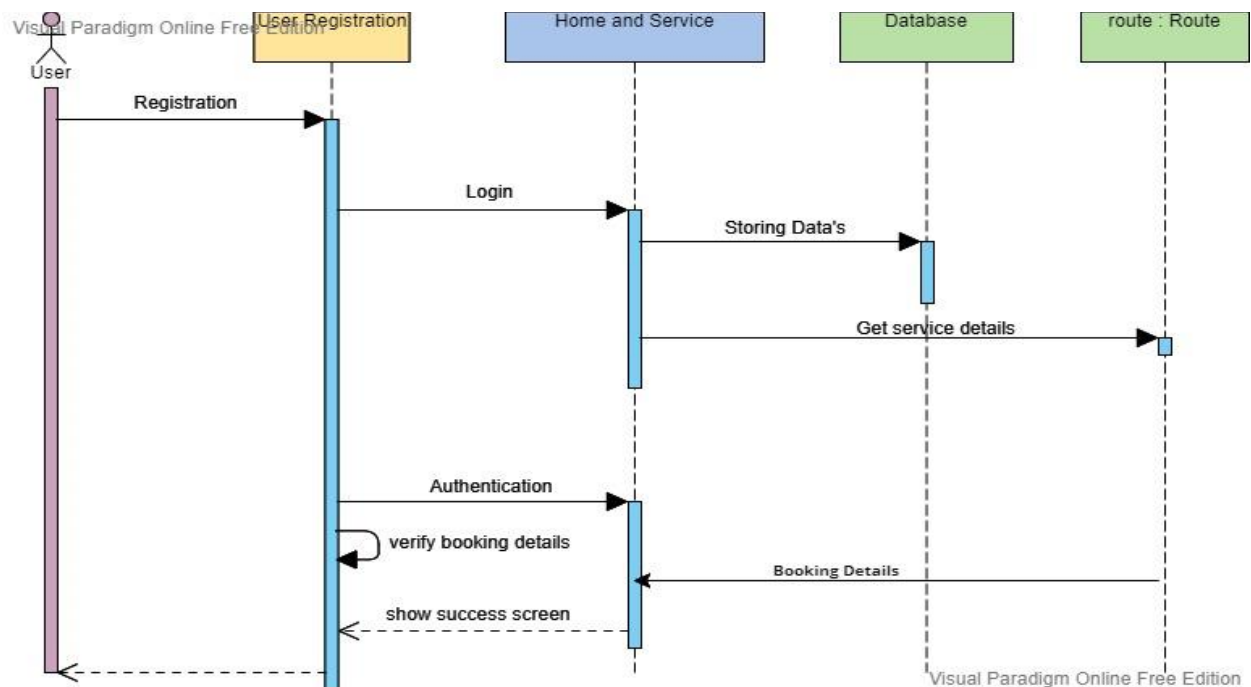
- For frontend framework, we have used **Angular** with needed components made with their necessary HTML, CSS and Typescripts.
- And for database we have used pure **MongoDB** where we stored the user data and retrieved to display them when needed like in case for booking, login and registration.

System Architecture

Use Case Diagram:



Sequence Diagram:



Source code of the algorithm

FRONTEND

Login.component.ts:

```

import { HttpClient } from '@angular/common/http';
import { Component, Input, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { RegistrationService } from '../registration.service';
import { Registration } from 'registration.model';
import { FormBuilder } from '@angular/forms';
import { BookingService } from '../booking.service';

@Component({
  selector: 'app-login',

```

```

    templateUrl: './login.component.html',
    styleUrls: ['./login.component.css']
  })
export class LoginComponent implements OnInit{
  currentUser?:Registration[];
  currentlogin: Registration = {};
  currentIndex = -1;
  email="";
  password="";

  constructor(private route:ActivatedRoute, private router: Router,private
  FormBuilder:FormBuilder, private registrationService:RegistrationService, private
  bookingService:BookingService) { }

  ngOnInit(): void {
    this.retrieveUsers();
  }

  retrieveUsers():void{
    this.registrationService.getAll()
    .subscribe({
      next:(data) =>{
        this.currentUser=data;
      },
      error:(e) => console.error(e)
    })
  }

  login():void{
    this.currentlogin={};

```

```
this.currentIndex=-1;
this.registrationService.logging(this.email)
.subscribe(res=>{
  const User=res.find((a:any)=>{
    return a.email===this.email && a.password===this.password;
  });

  if (User) {
    alert("Login Successful")
    this.router.navigate(['home'])
  } else {
    alert("User Not Found")
  }
});

}
}
```

Register.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Registration } from 'registration.model';
import { RegistrationService } from '../registration.service';
```

```
@Component({
  selector: 'app-registration',
  templateUrl: './registration.component.html',
  styleUrls: ['./registration.component.css']
})
export class RegistrationComponent{
  registration:Registration={
    fullname:'',
    email:'',
    password:'',
  }
  registered=false;
  constructor(private router:Router,private registrationService:RegistrationService) { }

  signUp(): void{
    const data={
      fullname:this.registration.fullname,
      email:this.registration.email,
      password:this.registration.password,
    };
    this.registrationService.create(data)
    .subscribe({
      next:(res)=>{
        this.registered=true;
        alert("Registration Successful")
      }
    })
  }
}
```



```
    },  
    error:(e) => console.error(e)  
  });  
  this.router.navigate(['login']);  
}  
}
```

Registration.model.ts

```
export class Registration{  
  id?:any;  
  fullname?:string;  
  email?: string;  
  password?:string;  
}
```

Registration.service.ts

```
import { Injectable } from '@angular/core';  
import { HttpClient, HttpHeaders } from '@angular/common/http';  
import { Observable } from 'rxjs';  
import { Registration } from 'registration.model';  
import { Router } from '@angular/router';
```

```
const baseUrl = 'http://localhost:8080/api/registration';
```

```

@Injectable({
  providedIn: 'root'
})
export class RegistrationService {

  constructor(private http: HttpClient,private router:Router) { }

  create(data: any): Observable<any> {
    return this.http.post(baseUrl, data);
  }
  getAll():Observable<Registration[]>{
    return this.http.get<Registration[]>(baseUrl)
  }
  logging(email:any):Observable<Registration[]>{
    return this.http.get<Registration[]>(` ${baseUrl}?email=${email}`);
  }
}

```

Booking.component.ts

```

import { Component, OnInit } from '@angular/core';
import{FormBuilder,FormGroup}from '@angular/forms'
import { Router } from '@angular/router';

```

```

import { bookingModel } from './booking.model';
import { BookingService } from '../booking.service';
@Component({
  selector: 'app-booking',
  templateUrl: './booking.component.html',
  styleUrls: ['./booking.component.css']
})
export class BookingComponent implements OnInit {

  formValue !: FormGroup;

  bookingModelobj : bookingModel = new bookingModel();

  constructor(private formbuilder: FormBuilder,private
  bookingService:BookingService,private router:Router) { }

  ngOnInit(): void {
    this.formValue = this.formbuilder.group({
      Vehicle:'',
      Vehiclemanufacturer:'',
      Vehiclemodel:'',
      Mobilenum:'',
      Address:'',
      Fueltype:'',
      Servicedate:'',
    })
  }
}

```

```

postEmployeeDetails():void{
    this.bookingModelobj.Vehicle=this.formValue.value.Vehicle;

this.bookingModelobj.Vehiclemanufacturer=this.formValue.value.Vehiclemanufacturer;
    this.bookingModelobj.Vehiclemodel=this.formValue.value.Vehiclemodel;
    this.bookingModelobj.Mobilenum=this.formValue.value.Mobilenum;
    this.bookingModelobj.Address=this.formValue.value.Address;
    this.bookingModelobj.Fueltype=this.formValue.value.Fueltype;
    this.bookingModelobj.Servicedate=this.formValue.value.Servicedate;
    const data=this.bookingModelobj
    this.bookingService.create(data)
    .subscribe({
        next:(res) =>{
            this.formValue.reset();
            alert("Booking Successful")
        },
        error:(e) => console.error(e)
    })
}

}

```

Booking.model.ts

```

export class bookingModel{

```

```
id?:any;
Vehicle?: string;
Vehiclemanufacturer?: string;
Vehiclemodel?: string;
Mobilenum?: string;
Address?: string;
Fueltype?:string;
Servicedate?:string;
}
```

Booking.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Router } from '@angular/router';
import { bookingModel } from '../booking/booking.model';
```

```
const baseUrl = 'http://localhost:7200/api/booking';
```

```
@Injectable({
  providedIn: 'root'
})
```

```
export class BookingService {
```

```

    constructor(private http: HttpClient,private router:Router) { }

    create(data: any): Observable<any> {
        return this.http.post(baseUrl, data);
    }

    getAll(): Observable<bookingModel[]> {
        return this.http.get<bookingModel[]>(baseUrl);
    }
}

```

Booking-list.component.ts

```

import { Component, OnInit } from '@angular/core';
import { bookingModel } from '../booking/booking.model';
import { BookingService } from '../booking.service';
@Component({
    selector: 'app-booking-list',
    templateUrl: './booking-list.component.html',
    styleUrls: ['./booking-list.component.css']
})
export class BookingListComponent implements OnInit {
    bookings?:bookingModel[];
    currentBooking : bookingModel={};
    constructor(private bookingService:BookingService) { }
}

```

```
ngOnInit(): void {  
  this.retrievebookings();  
}  
retrievebookings():void{  
  this.bookingService.getAll()  
    .subscribe({  
    next: (data) => {  
      this.bookings = data;  
    },  
    error: (e) => console.error(e)  
  });  
}  
}
```

BACKEND

BOOKING-SERVER

SERVER.JS

```
const express = require("express");
```

```
const cors = require("cors");
```

```
const app = express();
```

```
var corsOptions = {
```

```
  origin: "http://localhost:4200"
```

```
};
```

```
app.use(cors(corsOptions));
```

```
app.use(express.json());
```

```
app.use(express.urlencoded({ extended: true }));
```

```
const db = require("./app/models");
```

```
db.mongoose
```

```
  .connect(db.url, {
```

```
    useNewUrlParser: true,
```

```
    useUnifiedTopology: true
```

```
  })
```

```
  .then(() => {
```

```
    console.log("Connected to the database!");
```

```
  })
```

```
  .catch(err => {
```

```
    console.log("Cannot connect to the database!", err);
```

```
    process.exit();
```

```
  });
```

```
app.get("/", (req, res) => {
```



```
res.json({ message: "Welcome to Registration." });  
});
```

```
require("../app/routes/booking.routes")(app);
```

```
const PORT = process.env.PORT || 7200;  
app.listen(PORT, () => {  
  console.log(`Server is running on port ${PORT}.`);  
});
```

BOOKING.CONTROLLER.JS

```
const { booking } = require("../models");  
const db = require("../models");  
const bookingModel = require("../models/booking.model");  
const Booking = db.booking;
```

```
exports.create = (req, res) => {  
  const booking = new Booking({  
    Vehicle:req.body.Vehicle,  
    Vehiclemanufacturer:req.body.Vehiclemanufacturer,  
    Vehiclemodel:req.body.Vehiclemodel,  
    Mobilenum:req.body.Mobilenum,  
    Address:req.body.Address,
```

```
Fueltype:req.body.Fueltype,  
Servicedate:req.body.Servicedate,  
});
```

booking

```
.save(booking)  
.then(data => {  
  res.send(data);  
})  
.catch(err => {  
  res.status(500).send({  
    message:  
      err.message || "Some error occurred while Booking the Service."  
  });  
});  
};  
  
exports.findAll = (req, res) => {  
  const Vehicle= req.query.Vehicle;  
  var condition = Vehicle ? { Vehicle: { $regex: new RegExp(Vehicle), $options: "i" } } : {};  
  
  Booking.find(condition)  
    .then(data => {  
      res.send(data);  
    })  
    .catch(err => {
```

```
res.status(500).send({  
  message:  
    err.message || "Some error occurred while booking."  
});  
});  
};
```

BOOKING.ROUTES.JS

```
module.exports = app => {  
  const booking = require("../controllers/booking.controller.js");  
  
  var router = require("express").Router();  
  
  router.post("/", booking.create);  
  router.get("/", booking.findAll);  
  app.use("/api/booking", router);  
};
```

REGISTRATION-LOGIN-SERVER

SERVER.JS

```
const express = require("express");  
const cors = require("cors");
```

```
const app = express();
```

```
var corsOptions = {  
  origin: "http://localhost:4200"  
};
```

```
app.use(cors(corsOptions));
```

```
app.use(express.json());
```

```
app.use(express.urlencoded({ extended: true }));
```

```
const db = require("./app/models");  
db.mongoose  
  .connect(db.url, {  
    useNewUrlParser: true,  
    useUnifiedTopology: true  
  })  
  .then(() => {  
    console.log("Connected to the database!");  
  })  
  .catch(err => {
```

```
    console.log("Cannot connect to the database!", err);  
    process.exit();  
  });
```

```
app.get("/", (req, res) => {  
  res.json({ message: "Welcome to Registration." });  
});
```

```
require("./app/routes/registration.routes")(app);
```

```
const PORT = process.env.PORT || 8080;  
app.listen(PORT, () => {  
  console.log(`Server is running on port ${PORT}.`);  
});
```

REGISTRATION.CONTROLLER.JS

```
const { registration } = require("../models");  
const db = require("../models");  
const registrationModel = require("../models/registration.model");  
const Registration = db.registration;
```

```
exports.create = (req, res) => {  
  const registration = new Registration({
```

```
    fullname: req.body.fullname,  
    email: req.body.email,  
    password: req.body.password,  
  });
```

registration

```
    .save(registration)  
    .then(data => {  
      res.send(data);  
    })  
    .catch(err => {  
      res.status(500).send({  
        message:  
          err.message || "Some error occurred while creating the Registration."  
      });  
    });  
  };  
  
exports.findAll = (req, res) => {  
  const email= req.query.email;  
  var condition = email ? { email: { $regex: new RegExp(email), $options: "i" } } : {};  
  
  Registration.find(condition)  
    .then(data => {  
      res.send(data);  
    })  
  }  
}
```

```
.catch(err => {  
  res.status(500).send({  
    message:  
      err.message || "Some error occurred while retrieving Users."  
  });  
});  
};
```

REGISTRATION.ROUTES.JS

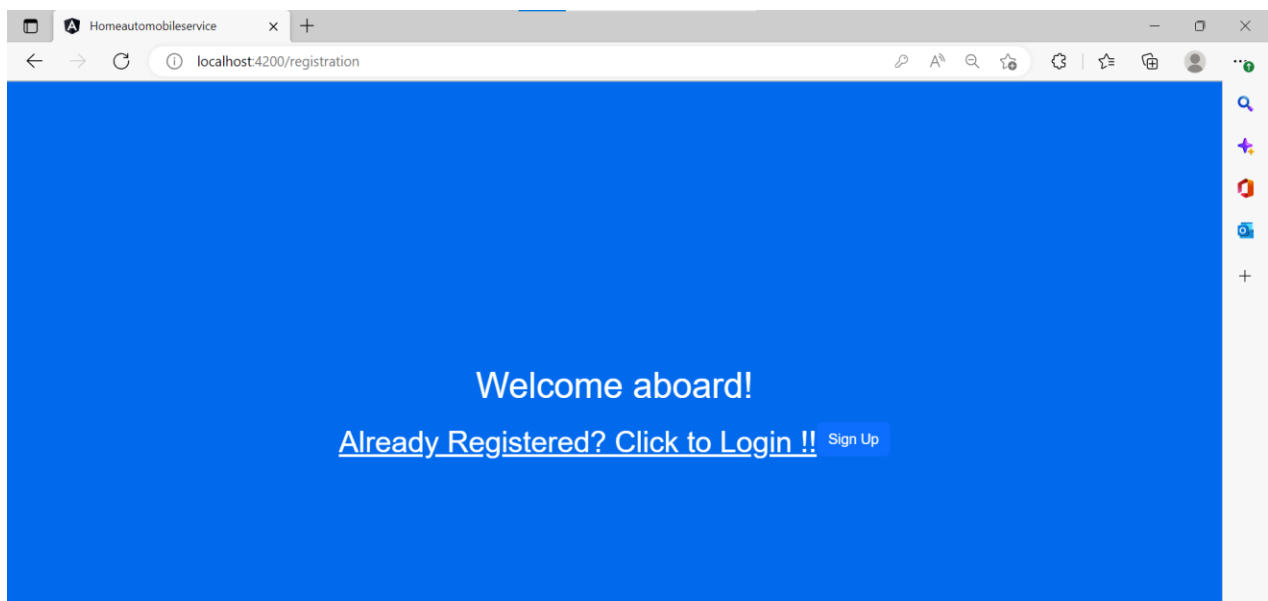
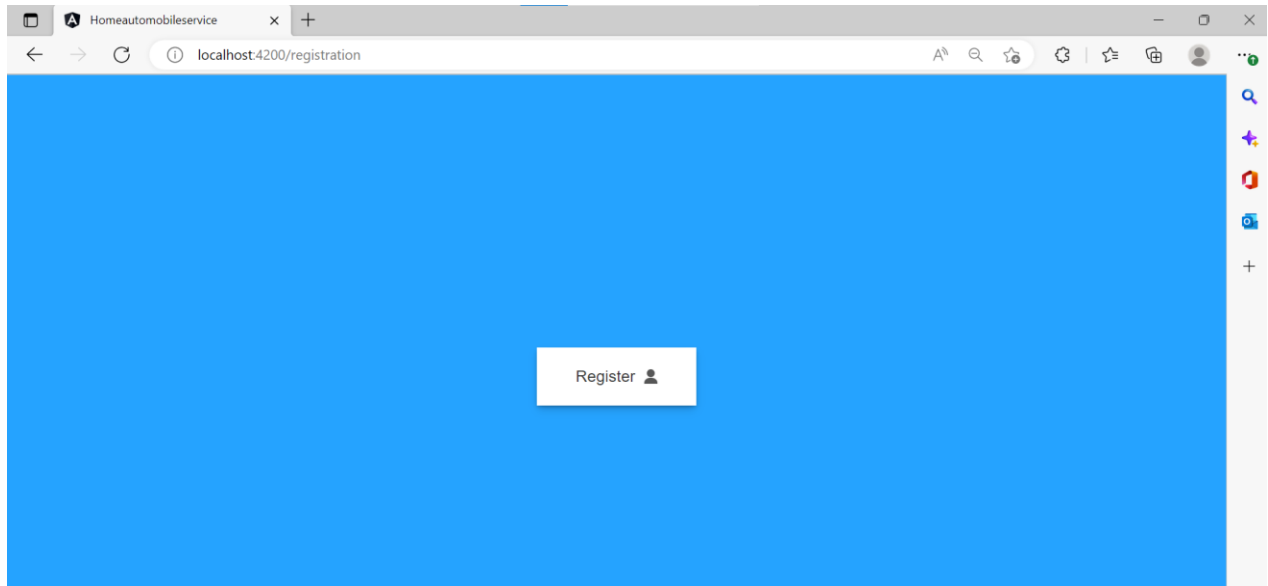
```
module.exports = app => {  
  const registration = require("../controllers/registration.controller.js");  
  
  var router = require("express").Router();  
  
  router.post("/", registration.create);  
  router.get("/", registration.findAll);  
  app.use("/api/registration", router);  
};
```

Code Repo Link

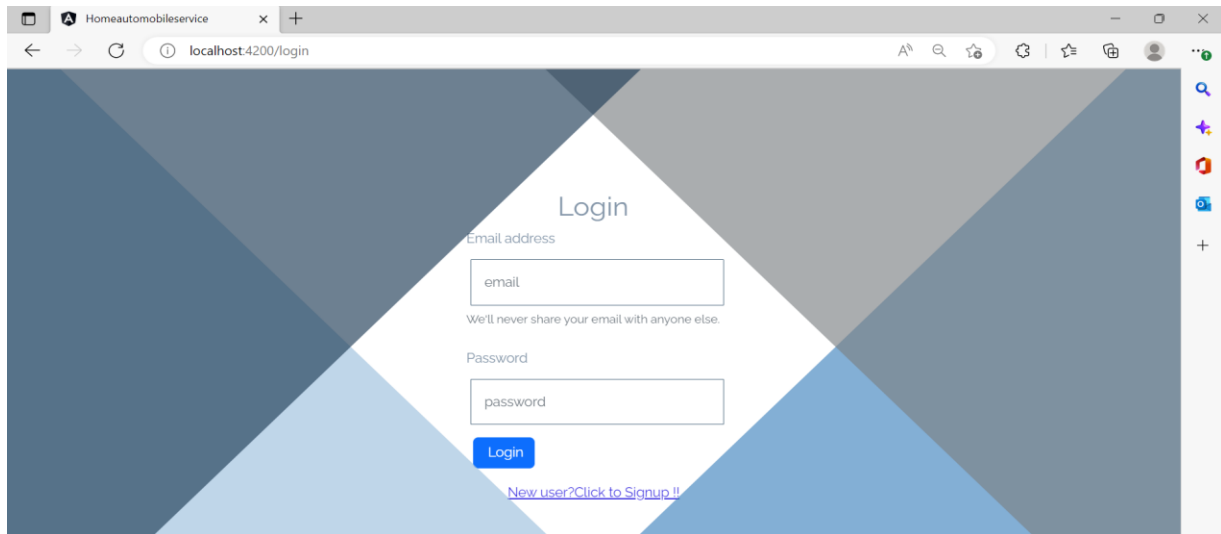
<https://github.com/vigneshrajan2001/Wheel-s-UP---Home-Automobile-Service.git>

Screenshots

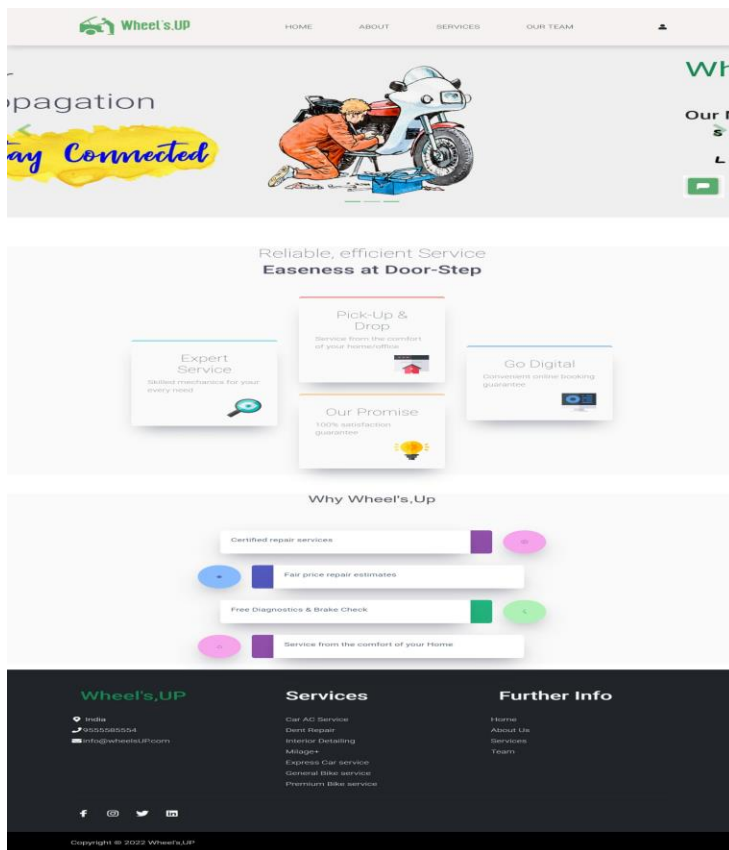
Registration Page:



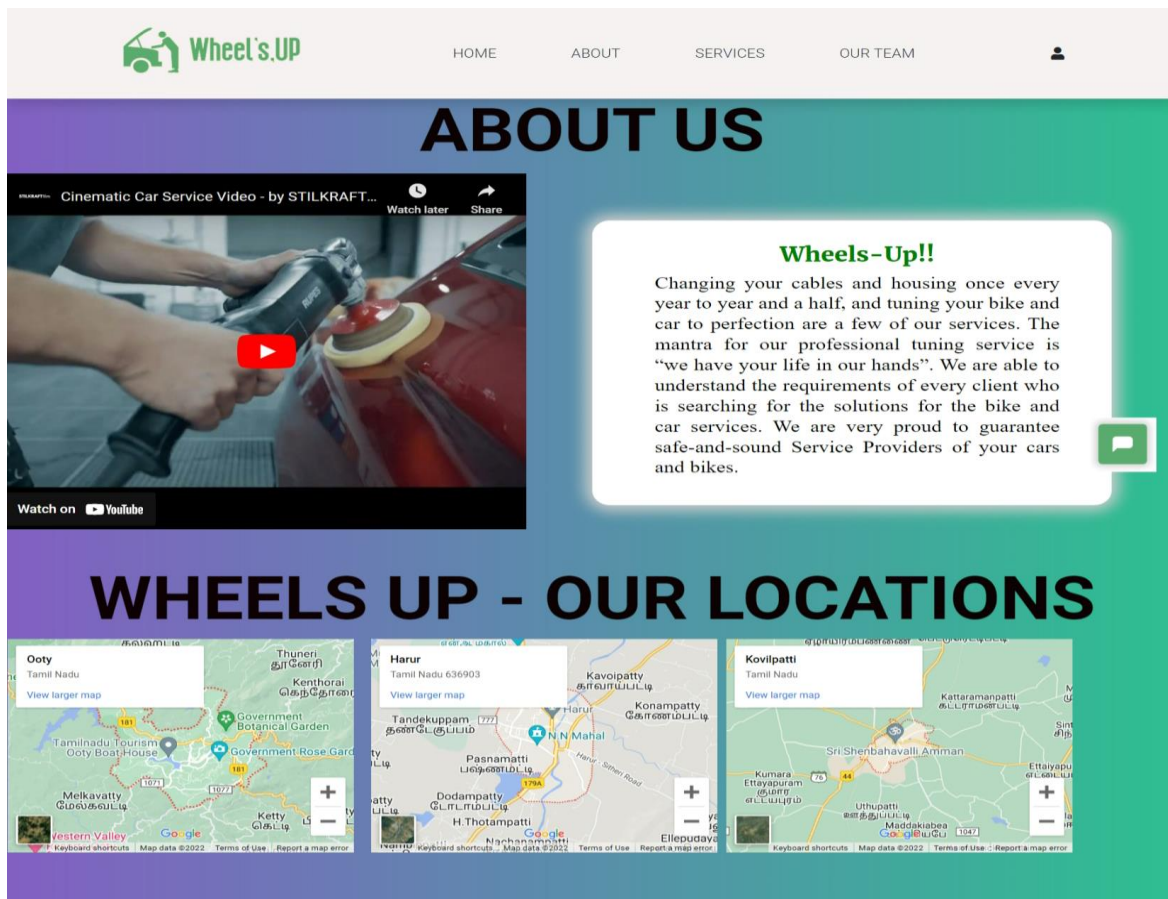
Login Page:



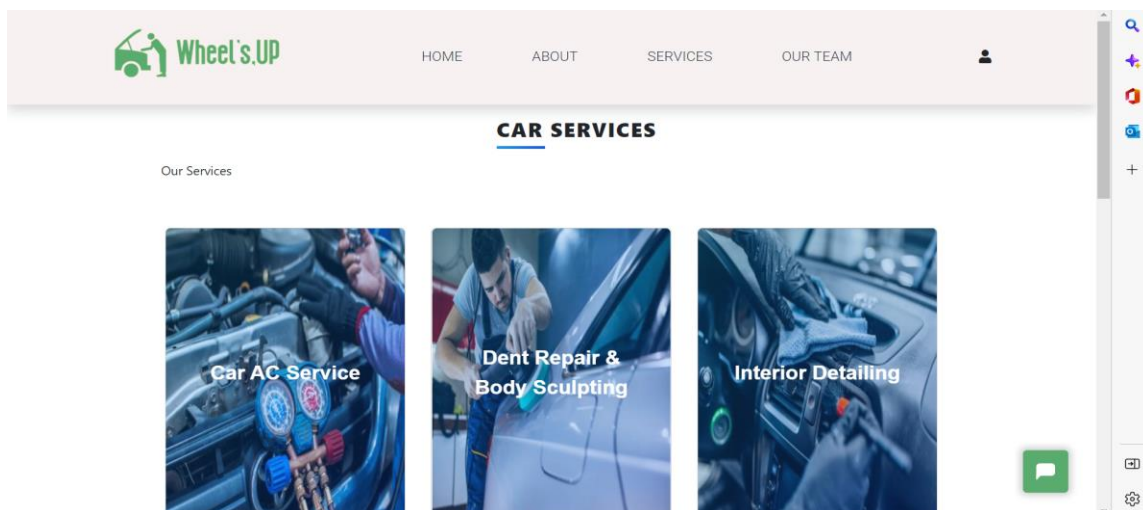
Home Page:





About-Us Page:



Services Page:



Booking Page:

HOMEABOUTSERVICESOUR TEAM

BOOK YOUR SERVICE

Select your Vehicle


Choose Vehicle Type


Select your vehicle Preference

Choose Manufacturer

Choose Model

Details

 Mobile No

 Address

Select your Fuel

Choose Fuel Type


Select Your Date

dd-mm-yyyy

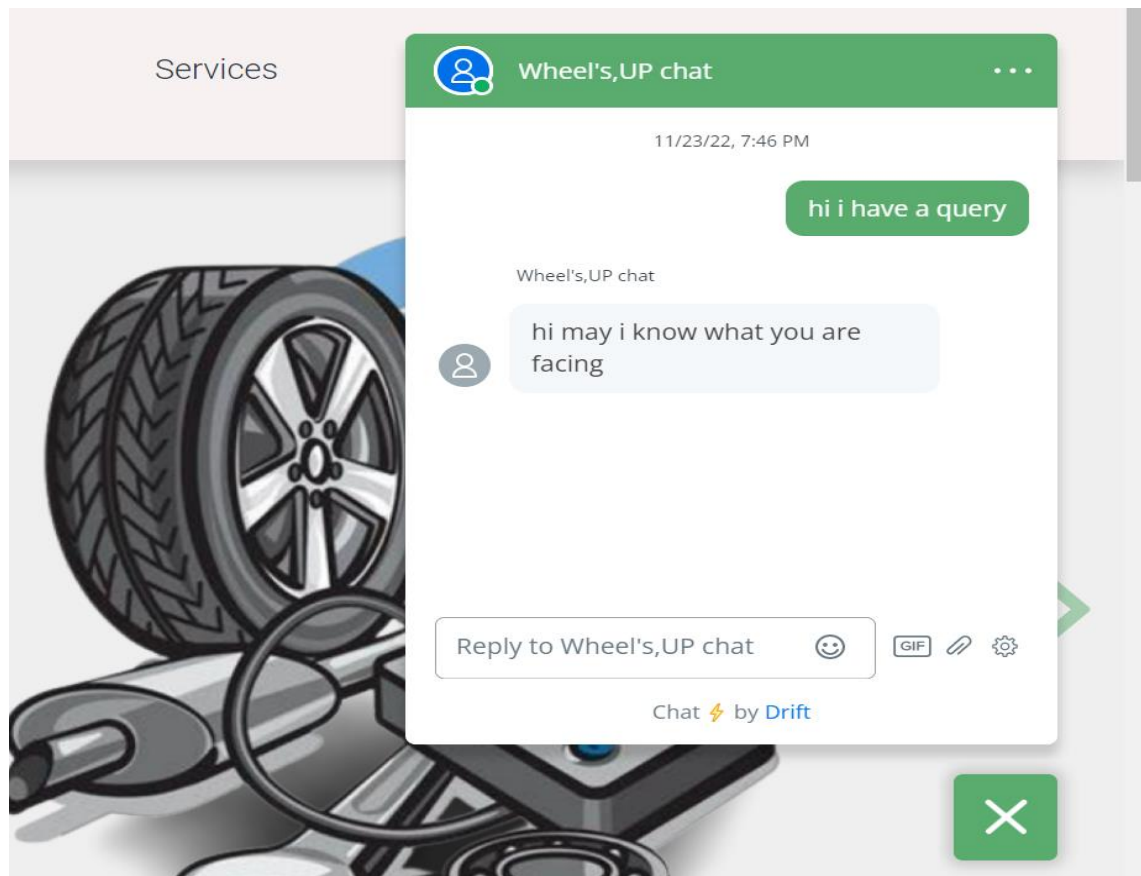
Terms and Conditions

☐ I accept the terms and conditions for signing up to this service, and hereby confirm I have read the privacy policy.

SUBMIT



Live Chat:



Analysis and Findings

Request and Response time with Remarks:

ACTIONS	-	DURATION	-	REMARKS
→ Request sent	-	0.13ms	-	Average
→ Server Response	-	2.28ms	-	Fair
→ Content Download	-	0.21ms	-	Good

USER-INTERFACE - RATING (rated by users)

→ Site Responsiveness	-	4/5
→ Site Features	-	3/5
→ Site Expectedness	-	5/5
→ User Satisfaction	-	4/5

IMPROVEMENTS THAT CAN BE DONE:

- A tracking feature can be added so that customer can know when their mechanic will arrive.
- 24/7 Assistance can be included which can be used during sudden breakdown or need of any assistance of repair during emergency.
- Online Payment can be added in future updates for convenient payment for users.

Conclusions

- Our project enables the users to book their services from the comfort of their home in a much faster and easier way.
- provide various kinds of services for both bike and car and also an interactive live chat for the customer to ask their queries.
- Through this we can establish a healthy relationship with customer by providing them services faster at their convenience anytime on their likes.
- This also helps in keeping tabs on databases which can be used as references for future queries and issues.

References

1. To know more visit: <https://freefrontend.com>, "CSS - Examples"
2. Visit: <https://www.britannica.com>, "Automobile | Facts, industry etc."
3. Visit: <https://www.otobots.com>, "Mobile Mechanics"
4. Gilles, Tim (2004). Automobile Service: Inspection, Maintenance, Repair. USA: Delmar Learning.pp. 16-23.
- 5." Independent garages and the motor Vehicle Block Exemption". UK Government. Retrieved 24 October 2012.