

Project Report
On
DAILY HOUSEHOLD SERVICES

By
Darji Jeelesh B.(CE-24) (19CEUBF016)
Divan Munaf S. (CE-27) (19CEUBG006)

B.Tech CE Semester - IV
Subject: Software Engineering

Guided by:
Prof.Pinkal c. Chauhan
Assistant Professor
Dept. of Comp. Engg.



Faculty of Technology
Department of Computer Engineering
Dharmsinh Desai University



**Department of Computer Engineering
Dharmsinh Desai University**

CERTIFICATE

**This is to certify that the practical / term work carried out in the subject of
System Design Practice and recorded in this journal is the
bonafide work of
DARJI JEELESH B.(CE-24) (19CEUBF016)
DIVAN MUNAF S. (CE-27) (19CEUBG006)
of B.Tech semester IV in the branch of Computer Engineering
during the academic year 2020-2021.**

Prof. Pinkal c. Chauhan
Assistant Professor,
Dept. of Computer Engg.,
Faculty of Technology
Dharmsinh Desai University, Nadiad

Dr. C. K. Bhensdadia,
Head, Dept. of Computer Engg.,
Faculty of Technology
Dharmsinh Desai University, Nadiad

➤ Table of Contents :-

1. Abstract.....	4
2. Introduction.....	5
2.1 Project Report : Brief Introduction	5
2.2 Technology and Tools Used	6
3. Software Requirement Specifications	7
3.1 Scope	7
3.2 System Functional Requirements	7
3.3 Other Non-Functional Requirements	14
4. Design.....	15
4.1 Use Case Diagram	15
4.2 Class Diagram	16
4.3 Sequence Diagram.....	17
4.4 Activity Diagram.....	19
4.5 Data Dictionary	21
5. Implementation Details	25
5.1 Modules	25
5.2 Functional prototypes	26
6. Testing	30
6.1 Testing Method.....	30
7. Screen-shots of the System	32
8. Conclusion	38
9. Limitations and Future Extensions of System	39
10. Bibliography	40

1.Abstract :

Now a days for any services like Plumbing, Electrical, Electronic, Mechanical, Machine Repairing & IT related Services, if any customer wants to use this type of services than they can go through a personal meeting or mobile call. It is difficult for customer to find any service in emergency at any time and place. So with this project we are going to develop website and android app which will help customers to find out solution for any problems related to Plumbing, Electrical, Electronic, and Mechanical Machine Repairing & IT related services. Our website will provide a platform for all kind of house hold services at any time and place. Our project will also provide the facilities like security, place navigation and also advertisement.

This system will overcomes the conventional system problems like a offline system, difficult to manage record, No time limit for service to be provided, Difficult to find proper service provider.

2. Introduction :

2.1 Brief Introduction :-

In this fast growing technology, we still have to take the appointment of person who solve the problems related to our daily life like plumbing related problem, mechanical problem, electrical problem, electronic problem, Mobile & Computer Related etc.

To take the appointment of service provider we have to call him or with the personal meeting we can meet him, and it is not sure that we get the appointment of the service provider at a time because there are many problems occur, like the service provider is busy at somewhere else or he is not present at his office when we go there or he wants heavy cost for fix the problem etc.

We are not getting any service on time and also not proper changes of services. It is also not secure in terms of safety concern.

To overcome these type of problem we are going to make our website where the people get appropriate result.

This website is very dynamic and very easy to understand. The interface of the website is very easy and anybody can easily work on it. This website can provide all the description and important information about the problem.

The Household service website is also very useful because the customer don't have to visit to service provider's office, he/she can easily book his/her order via this application and he/she can also pay the payment After Service had been Served. So he/she can book order without any kind of disturbance. It will provide security for the customer.

To make this application work successfully we have used some latest technology such as Django Framework as the platform and MYSQL as the database management environment. we had use github as a version control system.

The UML diagram has been drawn which is useful to display the flow of the process throughout the system so even an inexperienced people can easily get the idea of the proposed system.

2.2 Tools/Technologies Used :-

✖ Technologies:

- ✓ Django
- ✓ Python
- ✓ MYSQL
- ✓ HTML
- ✓ CSS

✖ Tools:

- ✓ GIT
- ✓ VISUAL STUDIO CODE
- ✓ PHPMYADMIN

✖ Plateform:

- ✓ Local development server

3. Software Requirement Specifications :

3.1 Product Scope :-

The scope of our project is to designing a complete environment to provide a safe and user friendly environment for online service booking. The main aim of the project is to provider an easy to use Web for services provided for customer.

We often get frustrated while taking the appointment of service provider because there the many problems are occur, like the service provider is busy art somewhere else or his not receiving our call or his cost is very high according to problem. So in this project we will remove this headache.

3.2 System Functional Requirements :-

➤ R.1. Account Management

❖ R 1.1 : Make New Account

Description : In order to use our site, user have to create account by providing required details in input boxes.

Input: Name of user, type of account, username,email/phone number and password.

Output: Message of successfully account created.

❖ R 1.2 : Login using existing account

Description : User can login using his/her username and password.

Input : Username and Password.

Output : user redirected to homepage of site.

❖ R 1.3 : Forgot Password

Description : If user forgot his/her password then using this function he/she can reset his/her password. In this function we use email/phone number of user to identify him/her. And then we provide link for change the password.

Input : Email or Phone number And new password.

Output: Message of success.

❖ R 1.4 : Change Password

Description : User can change his/her password by providing old password and new password.

Input : Old password and New password.

Output : Message of success.

❖ R 1.5 :- Update the Account Details :-

Description : If User wants to Update His/Her Account Details Then They can Do it by using this Function.

Status :- User must be Successfully Logged in.

- R.1.5.1 :- Display current Details

Input : Choose Update Option.

Output : Shows a Currently Stored Details.

- R.1.5.2 :- Update current Details

Input : Change needed details.

Output : Updated Details with a Confirmation of Changes.

❖ R 1.6 : Delete Account

Description : User can delete his/her account from site. User need to select “delete account” option and then enter password to conform his identity. Then his account will be deleted.

Input : Selection of “delete account” option and password.

Output : User redirected to login page.

✚ R.2. Service Management :-

❖ R.2.1 :- Login Using Existing Account :

Description : Service Provider can Login into the system using his/her Credentials.

Input : Enter Username & Password.

Output : Show Pop-up for Successfully Logged in.

Processing :- Validate Credentials in Database.

❖ R.2.2 :- Display current Details of all Services_:

Description : All of The Details about Respective Service provider & Services Displayed.

Input : User Selection.

Output : Details about Service Provider & Services.

❖ R.2.3 :- Add the Service _:

Description : Service Provider can Enter the Service details Which He/She Wants to Provide to the Customers. Service Details Contains Type of Service, Description, etc.

Status :- Service Provider must be Successfully Logged in.

Input : Enter Valid Service details.

Output : Show Details Entered by a User & Prompted a Confirmation

Message.

❖ R.2.4 :- Update the Service Details :-

Description : Service Provider can Update the Service details According to his/her Requirement.

Status :- Service Provider must be Successfully Logged in.

○ R.2.4.1 :- Display current Details

Input : Choose Update Option.

Output : Shows a Currently Stored Details.

○ R.2.4.2 :- Update current Details

Input : Change needed details.

Output : Updated Details with a Confirmation of Changes.

❖ R.2.5 :- Delete the Service :-

Description : Service Provider can Remove the Service which is no Longer Available to his/her Firm.

Status :- Service Provider must be Successfully Logged in.

Input : Choose Delete Service option.

Output : Prompted a Confirmation Message to remove the service.

+ R.3. Order Management :-

❖ R 3.1: Search for Service Provider

Description : User can search for service based on his/her requirement. In this function he/she need to provide some details like type of service and address where service is needed. Based on these details, list of service providers will be displayed in non-decreasing order of their rating.

Input : Type of the service and address.

Output : List of service providers in non-decreasing order of their rating will be displayed.

❖ R 3.2: View details of Service Provider

Description : From List of the service providers displayed in “Search” function user may select one of them using this function to know the details of that Service provider.

Input : Selection of Provider.

Output : Details of a service provider will be displayed.

❖ R 3.3: Place the Order

Description : From searching and viewing information of service providers, User may place the order for service to best suitable service provider.

○ R 3.3.1: Login

Description : User have to login for placing the order using login details.

Input : Username and Password.

○ R 3.3.2: Selection of Service Provider

Description : For placing the order user must select one of the service provider of his/her choice.

Input : Selection of service provider.

Output : User redirected to the page where he/she can place order.

○ R 3.3.3: Finalize the order

Description : In this function user finalize his order by proving OTP sent to his email id/phone number.

Input : Selection of mode (phone number/email) and OTP.

Output : The list of people who are waiting including him/her.

❖ R 3.4: Cancel the Order

Description : If user want to cancel his/her order before worker of service provider reaches his/her address he/she can do that using this function.

- R 3.4.1: Login

Description : User have to login for placing the order using login details.

Input : Username and Password.

- R 3.4.2 : Check availability of cancel option

Description : If Service provider is still busy with other orders then cancel option is available.

- R 3.4.3 : Proceed for cancellation

Description : User can cancel his/her order by providing OTP.

Input : click on cancel button and providing OTP.

Output : Success message.

✚ R.4. Administrator Management :-

- ❖ R.4.1 :- Login Using Existing Account :

Description : Administrator can Login into the system using his/her Credentials.

Input : Enter Username & Password.

Output : Show Pop-up for Successfully Logged in.

Processing :- Validate admin Credentials in Database.

- ❖ R.4.2 :- Display Details of all End Users :

Description : Administrator can view all of the details about customer & Service Provider. Admin can also view all Transaction Details between Users & Service Provider.

Input : Choose Show Details Option.

Output : Details about Service Provider & Users & between Them.

❖ R.4.3 :- Show TOP rated Service Provider :-

Description : Administrator can able to Display Top rated Service Provider Based on Rating & Feedback Giver by the Customer to Respective Provider.

Input : Manage List of Service Provider According Rating & Preference.

Output : Updated Details will be Shown to the All Customers.

❖ R.4.4 :- Remove or Notify Service Provider :-

Description : As Administrator can able to view Details about Service Provider, Admin can also Remove Account or send Notification to that firm if provider obtain sufficiently low Rating or excessive Report Abuse.

Input : Enter respective service Provider & choose actions to Perform.

Output : Updated are convey to Service Provider.

Processing : Updates are Reflected in a Database.

➤ **3.3 Other Nonfunctional Requirements :-**

1. Performance

The system must be interactive and must not involve long delays. Though in case of opening the app components or loading the page the system shows the delays less than 2 seconds.

2. Safety

The users' data is highly personal. The system has authorization to avoid any un-authorized access to user's private data.

3. Reliability

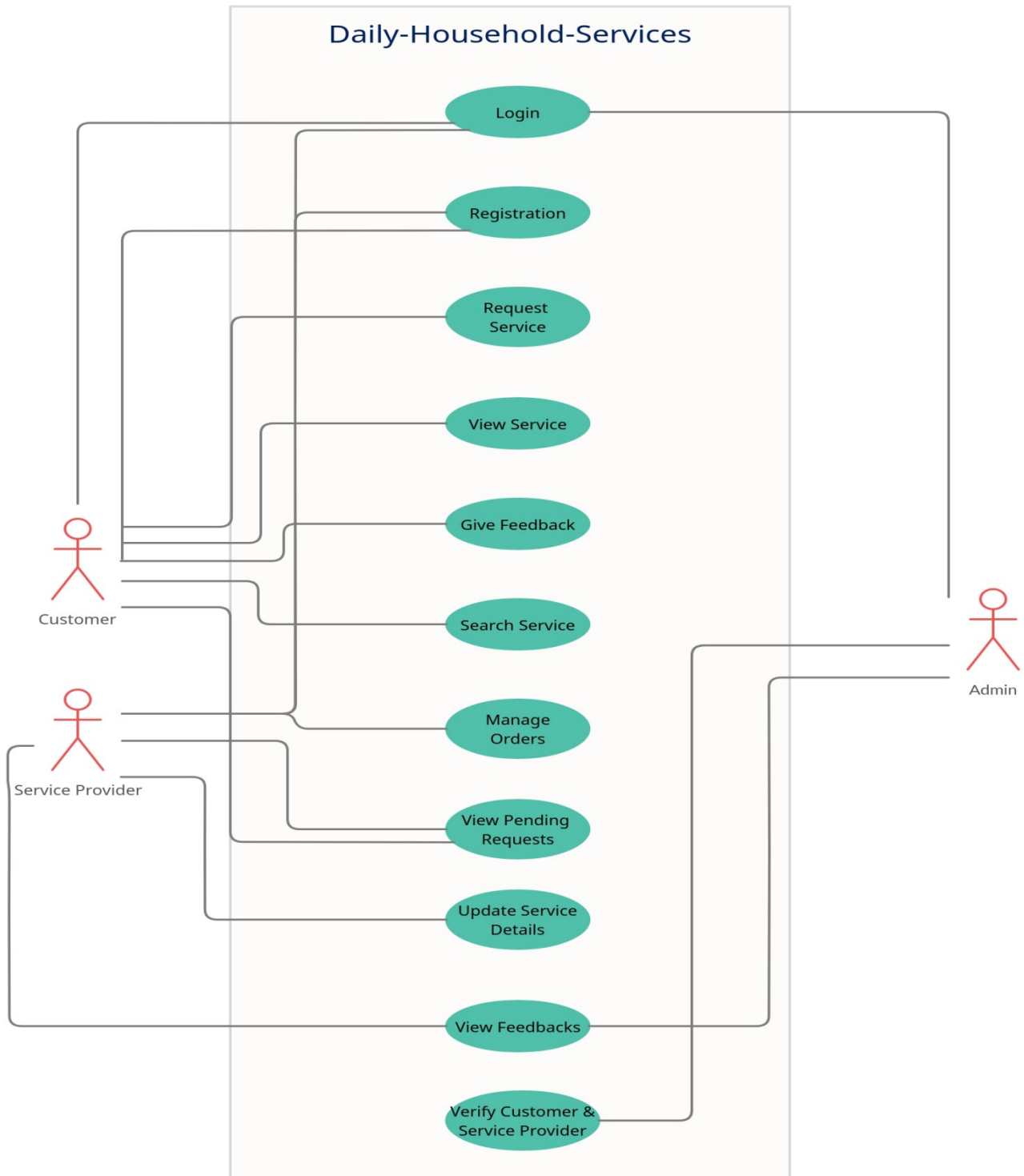
As the system has personal data, its reliability is the major factor for consideration.

4. Database

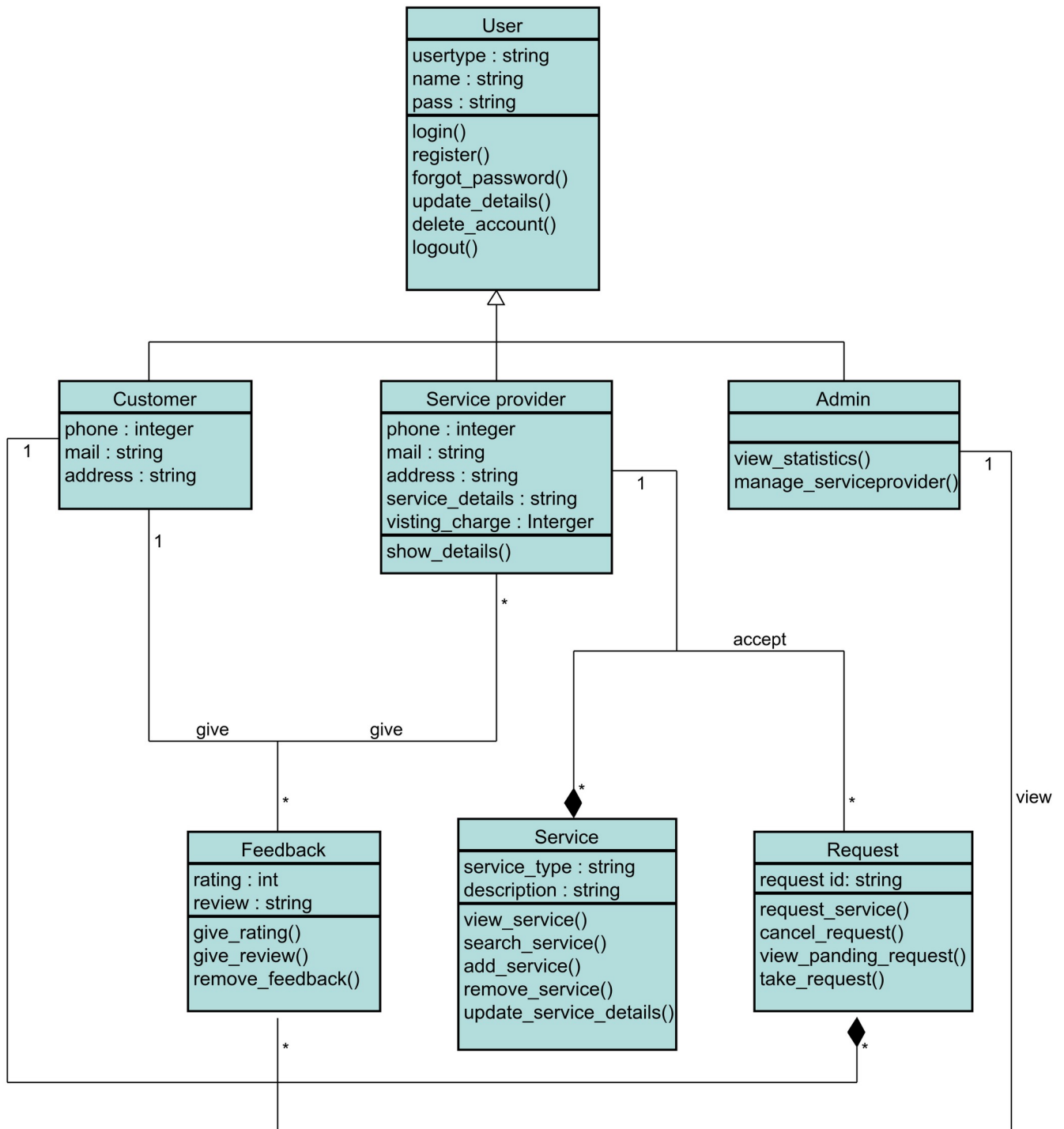
System requires to access user submissions, versions, reviews and profile data fast to maintain the performance.

4. Design :

4.1 Use Case Diagram :-

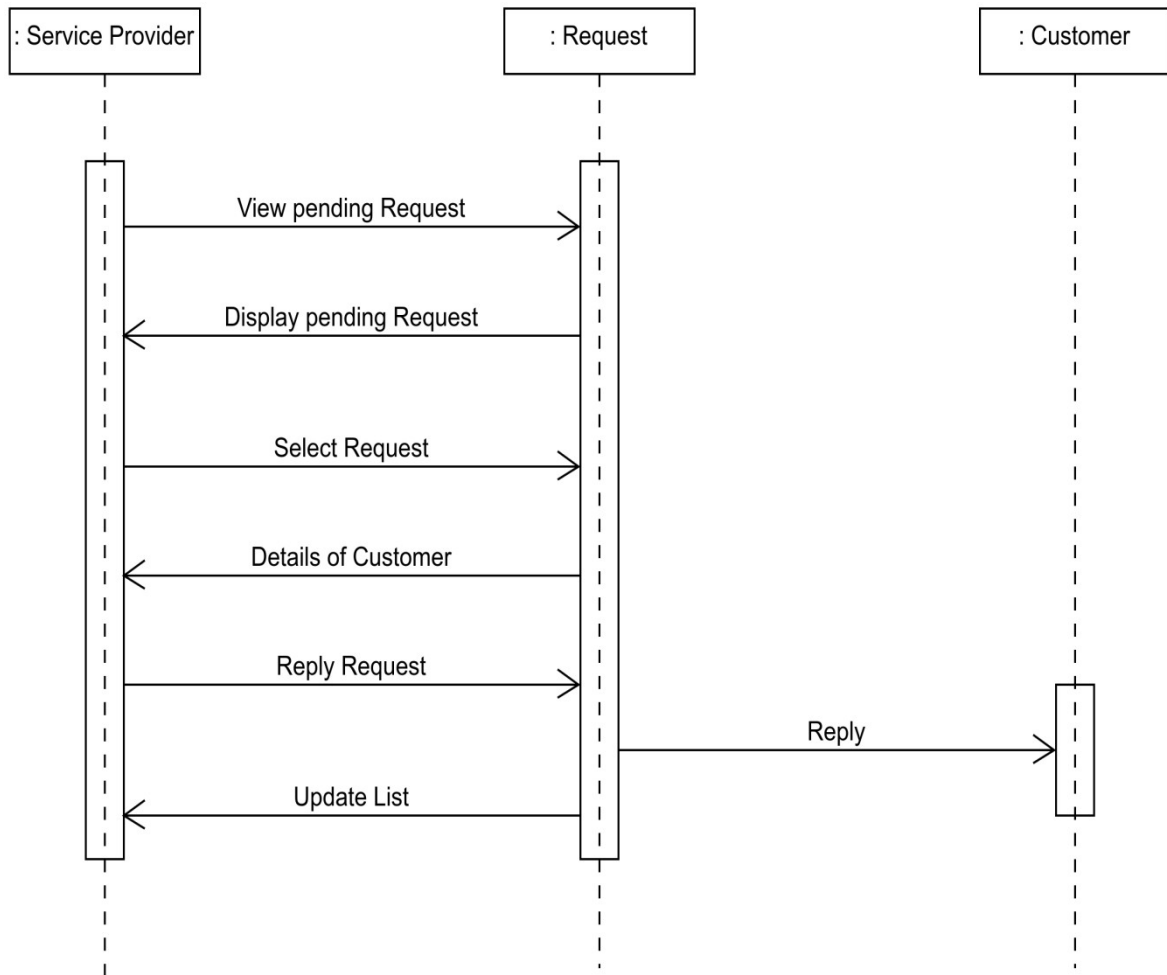


4.2 Class Diagram :-



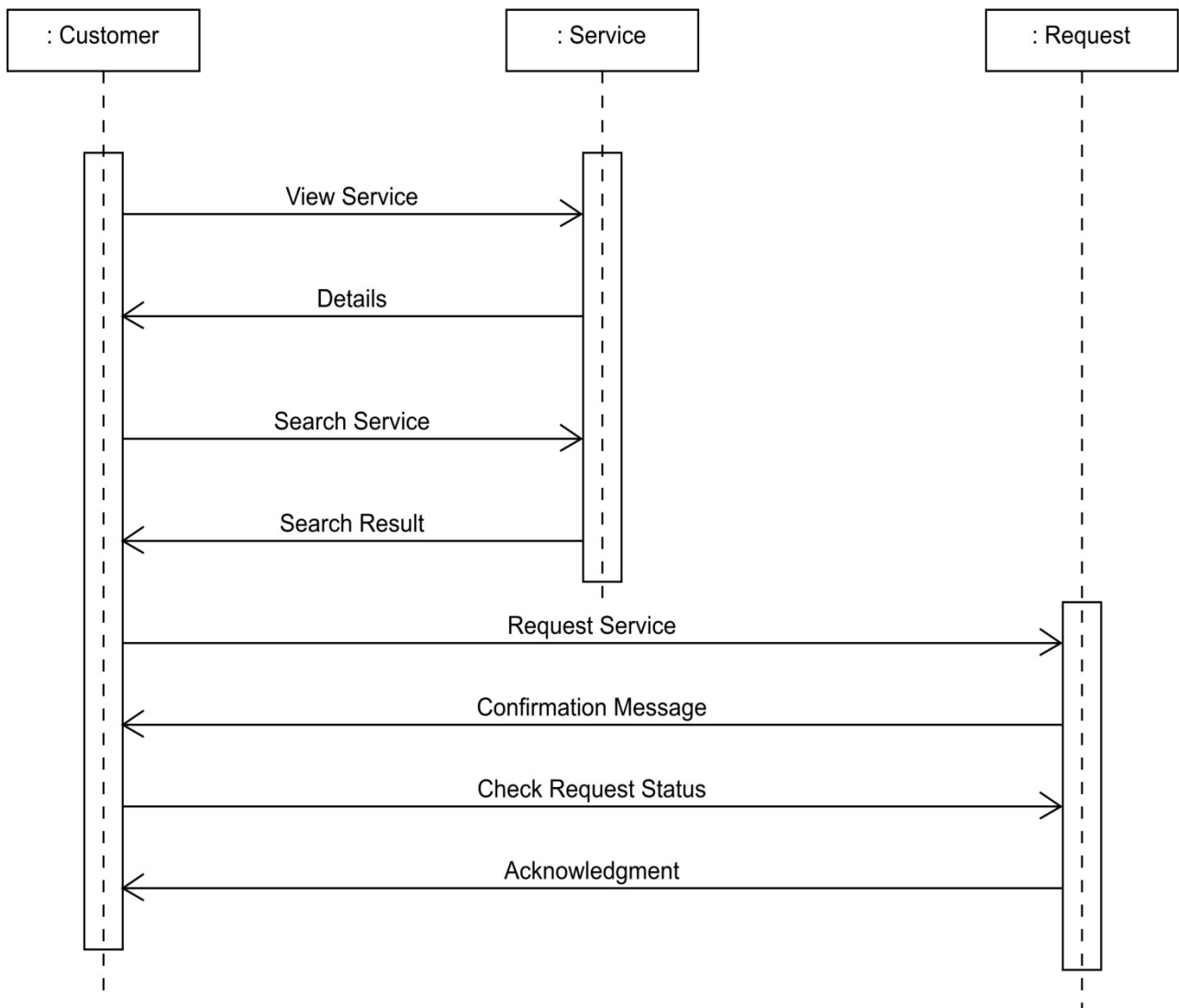
4.3 Sequence Diagram :-

I.)



[order]

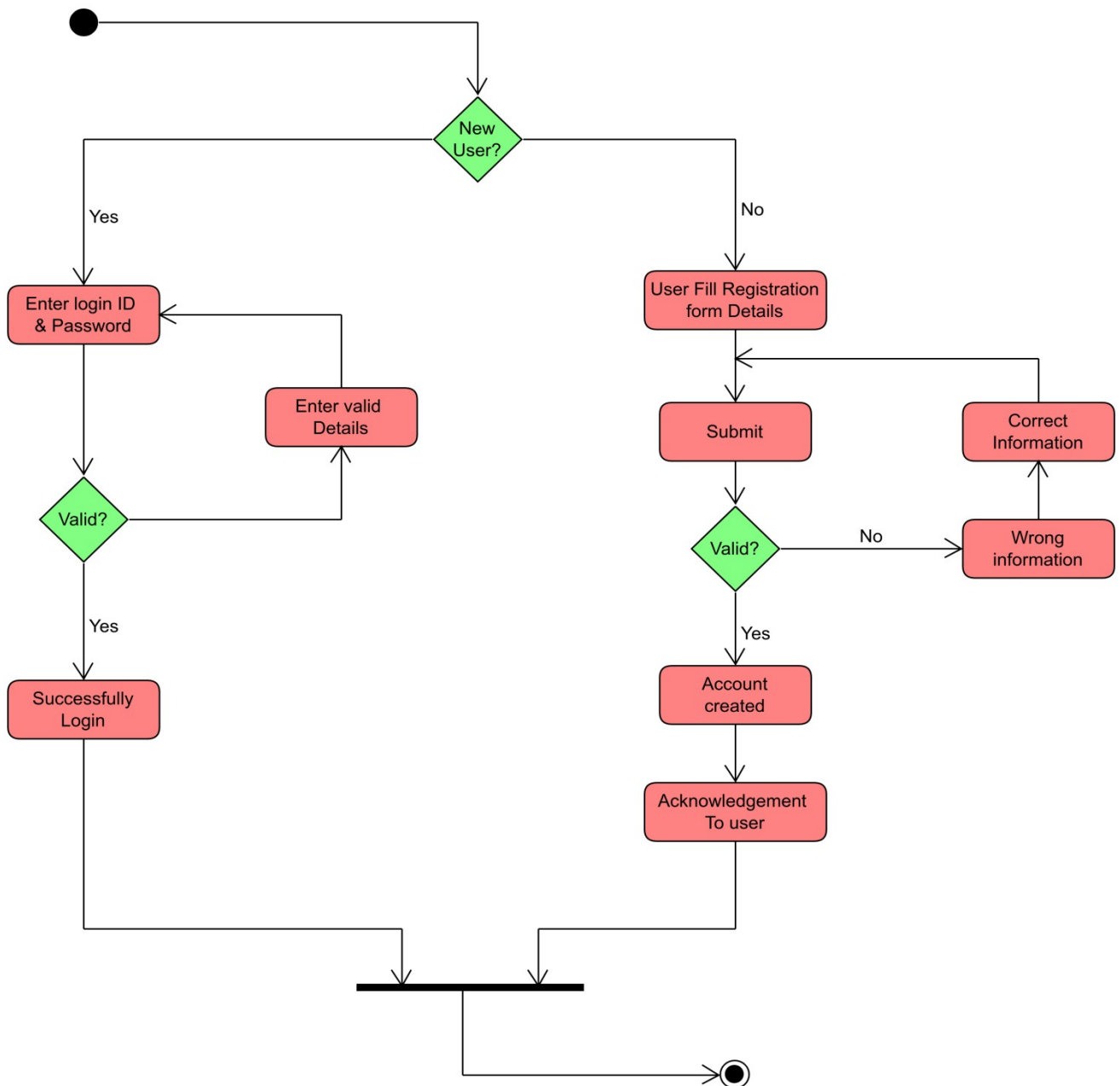
II.)



[Request Service]

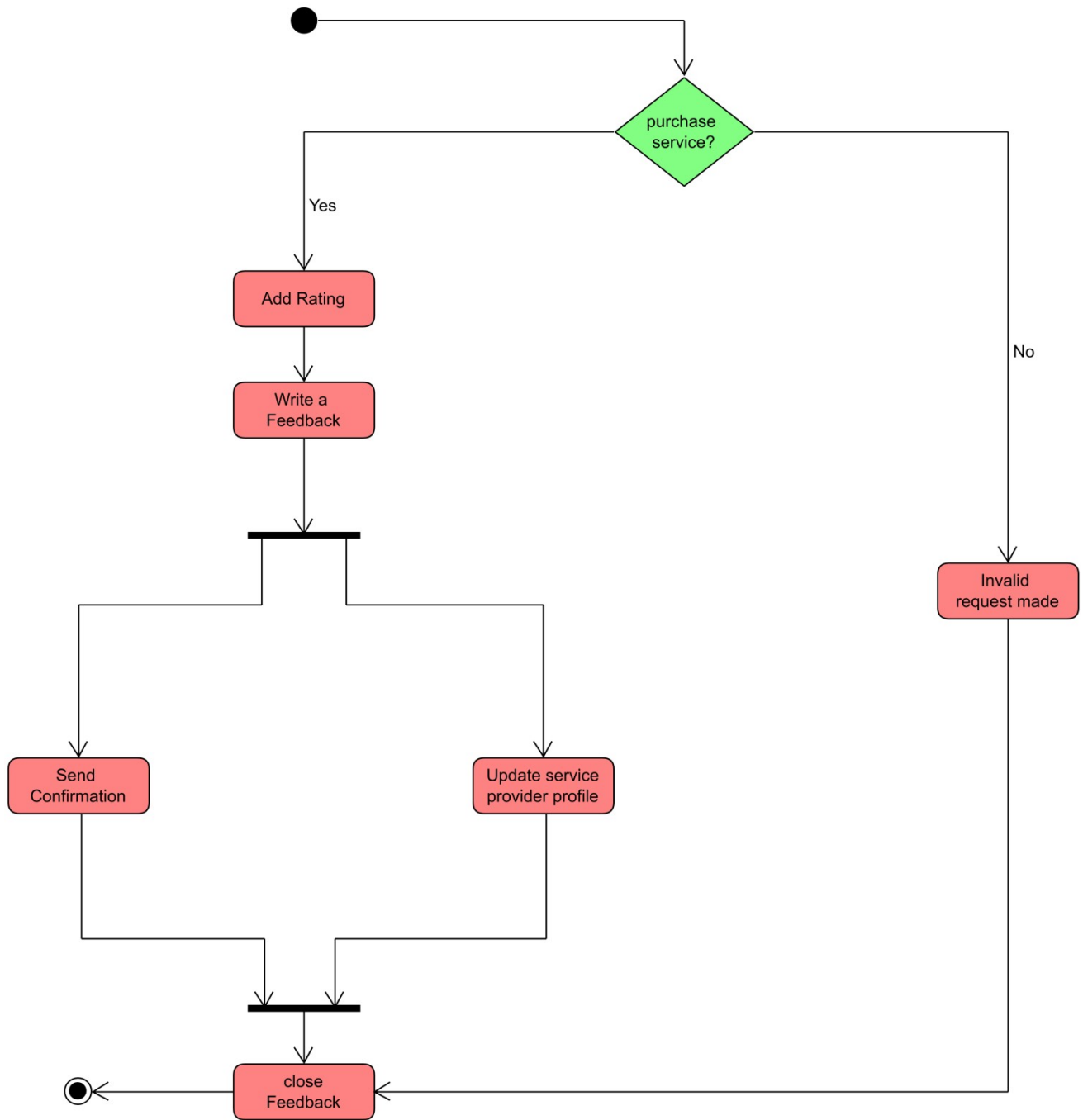
4.3 Activity Diagram :-

I.)



[Login]

II.)



[Feedback]

4.2 Data Dictionary :-

User

Sr no.	FieldUser Name	Data Type	Width	Required	Unique	PF/FK	Referred Table	Description
1	Id	int	11	yes	yes	Yes	-	Auto Increment
2	username	varchar	150	Yes	Yes	no	-	
3	email	varchar	254	yes	yes	no	-	
4	password	varchar	128	yes	no	no	-	
5	last_login	datetime	16	yes	No	No	-	
6	is_superuser	tinyint	1	yes	-	-	-	
7	first_name	varchar	30	Yes	no	no	-	-
8	last_name	varchar	150	Yes	no	No	-	-
9	date_joined	datetime	6	Yes	No	No	-	
10	is_active	tinyint	1	Yes	NO	No	-	-

Category

Sr no.	Field Name	Data Type	Width	Required	Unique	PF/FK	Referred Table	Description
1	Id	int	11	Yes	Yes	Yes	-	Auto Increment
2	Category	varchar	50	Yes	No	No	-	

Service Provider

Sr no.	Field Name	Data Type	Width	Required	Unique	PF/FK	Referred Table	Description
1	Id	Int	11	Yes	Yes	Yes	-	Auto Increment
2	username	varchar	100	Yes	Yes	No		
3	Password	varchar	10	Yes	No	No		
4	First_name	varchar	100	Yes	No	No		
5	Last_name	varchar	100	Yes	No	No		
6	Address	varchar	255	Yes	No	No		
7	City	Varchar	100	Yes	No	No		
8	Mobile_no	varchar	10	Yes	No	No		
9	Email	Email	254	Yes	No	No		
10	User_Type	varchar	5	Yes	No	No		
11	Service_rate	Int	11	Yes	No	No		
12	S_license	varchar	50	Yes	No	No		
13	Profile_description	varchar	255	Yes	No	No		
14	Cat_id_id	int	11	Yes	Yes	Yes	Category	

Customer

Sr no.	Field Name	Data Type	Width	Required	Unique	PF/FK	Referred Table	Description
1	Id	Int	11	Yes	Yes	Yes	-	Auto Increment
2	username	varchar	100	Yes	Yes	No		
3	Password	varchar	10	Yes	No	No		
4	First_name	varchar	100	Yes	No	No		
5	Last_name	varchar	100	Yes	No	No		
6	Address	varchar	255	Yes	No	No		
7	City	Varchar	100	Yes	No	No		
8	Mobile_no	varchar	10	Yes	No	No		
9	Email	Email	254	Yes	No	No		
10	User_Type	varchar	5	Yes	No	No		

Order Service								
Sr no.	Field Name	Data Type	Width	Required	Unique	PF/FK	Referred Table	Description
1	Id	Int	11	Yes	Yes	Yes	-	Auto Increment
2	C_id_id	varchar	100	Yes	Yes	Yes	Customer	Referred Field=Username
3	S_id_id	varchar	100	Yes	Yes	Yes	Service Provider	Referred Field=Username
4	Order_status	varchar	10	Yes	No	No		
5	City	varchar	50	Yes	No	No		
6	Address	varchar	255	Yes	No	No		
7	Email	Email	254	Yes	Yes	No		
8	Mobile_No	varchar	10	Yes	Yes	No		

5. Implementation Details :

5.1 Modules :-

1 . Registration Module

Basic Information of user is taken and stored in database.

2 . Login Module

Users are able to login themselves. System logs user in, then and only then user can use other functionalities of system.

3 . Category Module

Admin Saves different category of service and Service Provider can Choose from those categories.

4 . Order Service

All Order related work done in this module.

Customer can order for service and cancel that order too.

Service Provider can decide what to do with his order Requests.

5 . Administration Module

Admin can manage Users of System And Categories.

5.2 Function Prototype :-

1 User Registration

```
def CustomerSignup(request):
    form = CustomerForm(request.POST)
    if form.is_valid():
        try :
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            email = form.cleaned_data.get('email')
            fname = form.cleaned_data.get('first_name')
            lname = form.cleaned_data.get('last_name')
            user = User.objects.create_user(username,email,password)
            user.first_name = fname
            user.last_name = lname
            user.save()
            form.save()
            return redirect("/loginmodule/login")
        except Exception :
            msg = "This user name has been taken."
            return render(request=request,template_name='cust_registration.html',context={'form':form,'msg':msg})
    else:
        return render(request=request,template_name='cust_registration.html',context={'form':form})

def SpSignup(request):
    form = ServiceProviderForm(request.POST)
    if form.is_valid():
        try :
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            email = form.cleaned_data.get('email')
            fname = form.cleaned_data.get('first_name')
            lname = form.cleaned_data.get('last_name')
            user = User.objects.create_user(username,email,password)
            user.first_name = fname
            user.last_name = lname
            user.save()
            form.save()
            return redirect("/loginmodule/login")
        except Exception :
            msg = "This user name has been taken."
            return render(request=request,template_name='sp_registration.html',context={'form':form,'msg':msg})
    else:
        return render(request=request,template_name='sp_registration.html',context={'form':form})
```

2 User Authentication

```
def auth_view(request):
    if request.method == 'POST':
        username = request.POST.get('username', '')
        password = request.POST.get('password', '')
        user_type = request.POST.get('user_type', '')
        user = auth.authenticate(username=username, password=password)
        if user is not None:
            if user_type == 'cust':
                try:
                    user1 = Customer.objects.get(username=username)
                except:
                    msg = "Wrong Username or Password"
                    return render(request, 'login.html', {'msg': msg})
                else:
                    auth.login(request, user)
                    return HttpResponseRedirect('/loginmodule/customer')
            elif user_type == 'sp':
                try:
                    user1 = ServiceProvider.objects.get(username=username)
                except:
                    auth.login(request, user)
                    msg = "Wrong Username or Password"
                    return render(request, 'login.html', {'msg': msg})
                else:
                    auth.login(request, user)
                    return HttpResponseRedirect('/loginmodule/serviceprovider')
            elif user.is_superuser == 1:
                auth.login(request, user)
                return HttpResponseRedirect('/loginmodule/admin')
            else:
                msg = "Wrong Username or Password"
                return render(request, 'login.html', {'msg': msg})
```

3 Order Service

```
@login_required(login_url='/loginmodule/login')
def add_order(request, id):
    if request.method == 'POST':
        city = request.POST.get('city', '')
        mobile = request.POST.get('mobile', '')
        address = request.POST.get('address', '')
        email = request.POST.get('email', '')
        s_id = id
        c_id = request.user.username
        order_status = "requested"
        ob = OrderService()
        ob.c_id = Customer.objects.get(username=c_id)
        ob.s_id = ServiceProvider.objects.get(username=s_id)
        ob.city = city
        ob.mobile_no = mobile
        ob.order_status = order_status
        ob.address = address
        ob.email = email
        ob.save()
        return redirect('/orderservice/show/cust')
    else:
        return render(request, 'OrderForm.html', {'id': id})
```

4 Update Order Status

```
return render(request, 'show_sp.html', {'sp': order})
@login_required(login_url='/loginmodule/login')
def update_status(request, id, status, type):
    if type == 'sp':
        order = OrderService.objects.get(id = id)
        if status == 'done':
            order.delete()
        else:
            order.order_status = status
            order.save()
        return redirect('/orderservice/show/sp')
    elif type == 'cust':
        order = OrderService.objects.get(id = id)
        order.order_status = status
        order.save()
        return redirect('/orderservice/show/cust')
    else:
        msg = 'Invalid Access'
        logout(request)
        return redirect('/loginmodule/login')
```

5 Add Service Category

```
@login_required(login_url='/loginmodule/login')
def add_service(request):
    if request.method == "POST":
        if request.user.is_superuser:
            try:
                form = ServiceForm(request.POST)
                if form.is_valid() :
                    form.save()
                    return redirect('/service/show')
                else :
                    raise FormError("not valid form")
            except FormError:
                render(request, 'error.html')
        else :
            logout(request)
            return redirect('/loginmodule/login')
    else:
        form = ServiceForm()
        return render(request, 'AddService.html', {'form':form})
```

6 Manage Users

```
@login_required(login_url='/loginmodule/login')
def show(request, type):
    if request.user.is_superuser:
        if type == 'cust':
            customers = Customer.objects.all()
            return render(request, 'ShowCustomers.html', {'customers': customers})
        elif type == 'sp':
            service_providers = ServiceProvider.objects.all()
            return render(request, 'ShowServiceProviders.html', {'sps': service_providers})
        else:
            msg = 'Invalid Access'
            logout(request)
            return render(request, 'login.html', {'msg': msg})
    else:
        msg = 'Invalid Access'
        logout(request)
        return render(request, 'login.html', {'msg': msg})

@login_required(login_url='/loginmodule/login')
def delete(request, type, id):
    if request.user.is_superuser:
        user = User.objects.get(username=id)
        user.delete()
        if type == 'cust' :
            cust_user = Customer.objects.get(username=id)
            cust_user.delete()
            return redirect('/administration/show/cust')
        elif type == 'sp' :
            sp_user = ServiceProvider.objects.get(username=id)
            sp_user.delete()
            return redirect('/administration/show/sp')
        else:
            msg = 'Invalid Access'
            logout(request)
            return render(request, 'login.html', {'msg': msg})
    else:
        msg = 'Invalid Access'
        logout(request)
        return render(request, 'login.html', {'msg': msg})
```

6. Testing :

Manual testing was performed in order to find and fix the bugs in development process.

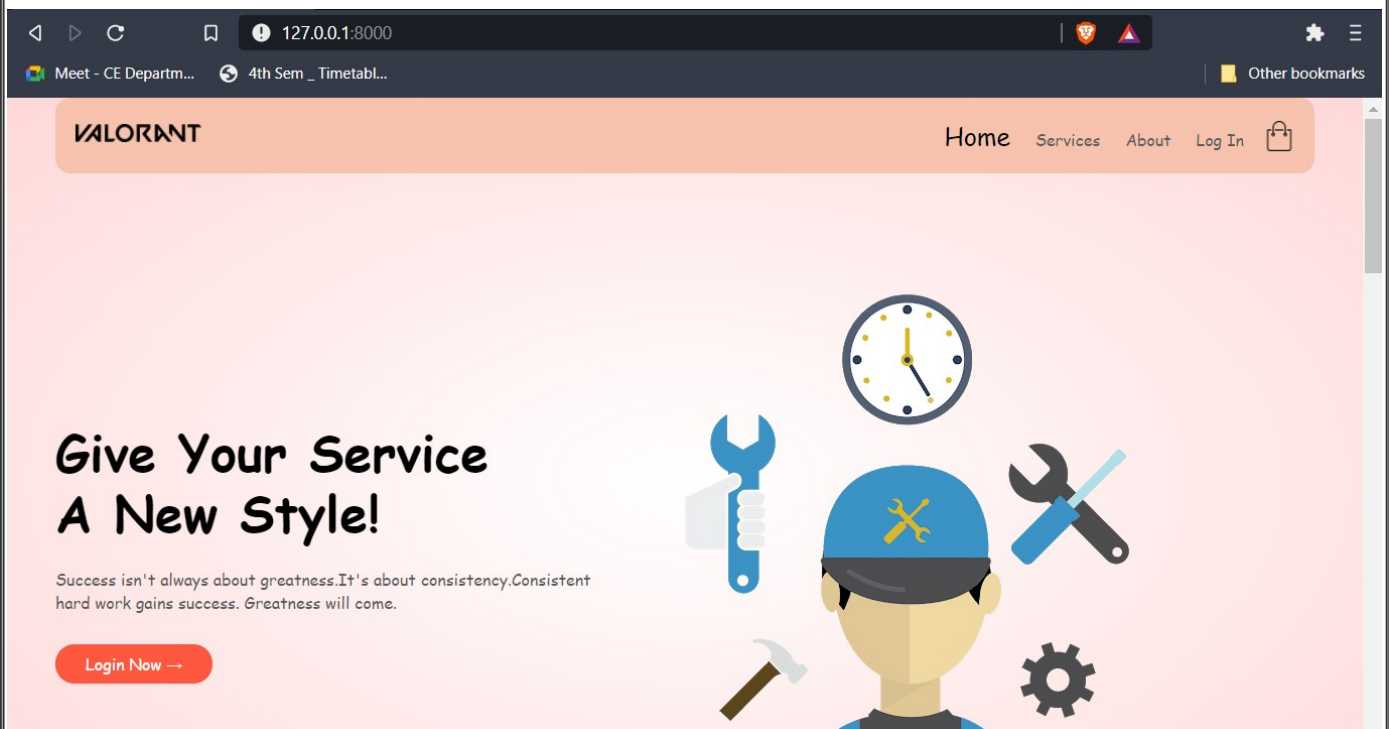
Testing Method: Manual Testing

Sr. No.	Test Scenario	Expected Result	Actual Result	Status
1	Login with incorrect credentials	User should not able to log in.	User is given a message. And redirected to login page.	Success
2	Login with correct credentials	User should be able to log in.	User is logged in and shown the appropriate page according to user type	Success
3	Validations on registration	User should not be allowed to enter incorrect email.	User is shown a message for any incorrect email.	Success
4	Registering with an existing Username	The system should show a message.	The system shows the message 'This Username has been taken'	Success
5	Access Other Functionality without login	User should not be able to use without login.	The System redirect user to login page.	Success
6	Order service from particular service provider	User should be able to order service and entry should be reflected to appropriate database tables.	Customer can order service and he can see his order and on other hand appropriate service provider is able to see order.	Success


7	Manage categories, service provider and customer	Admin should be able to manage category and delete service provider and customer	Admin can add/delete category and he can also delete service provider and customer	Success
---	--	--	--	---------

7. Screen-shots :

Home Page



Login Page

VALORANT[Home](#)[Services](#)[About](#)[Log In](#)


Sign Up

User Name


Enter password

Select User Type....


Log In





Show Services Page


VALORANT[Home](#)**Services**[About](#)[Logout](#)


Featured Services


Plumber


Electrician


Carpenter


IT-Repair



Mechanic

127.0.0.1:8000/orderservice/show_sp/electrician

Show Order >>

Show Service Provider

VALORANT

Home Services About Logout 


Service Providers

Name	mobile_no	Address	City	Email	Service Rate	Profile Description	Actions
Maresh Solanki	8192536592	Mareshana	Vishnagar	mareshbhai007@gmail.com	0	we provide service of plumbing	Order

[Back <](#)

Show order page

VALORANT

Home Services About Logout 

List of all Orders

Order ID	City	Mobile_no	Address	Email	status	Action
1	Vishnagar	8141911217	near Panchayat , kadvasan	mmdivan@gmail.com	requested	Cancel

[< Back](#)

Service Provider Home

VALORANT

Home Orders About Logout 

Welcome Mahesh_101 !

It's the page for only Service Provides for manage orders which they get from users & mark them as per Provides Details
You can Accept or Cance Order & after fullfiled users Requirement close the Order.

See Order →



Orders

VALORANT

Home Orders About Logout 

List of all Orders

Order ID	City	Mobile_no	Address	Status	Action
1	Vishnagar	8141911217	near Panchayat , kadvasan	requested	<button>Reject</button> <button>Accept</button>

< Back

Admin Home



Manage Service Provider

VALORANTHome About Logout

List of all Service Provider


S-ID	Username	Name	City	Address	Mobile No	Email	Service Rate	Service License	Profile Description	Actions
1	Mahesh_101	Mahesh Solanki	Vishnagar	Maheshana	8192536592	maheshbhai007@gmail.com	0	plumbing10001	we provide service of plumbing	Delete
2	sharman_123	Sharman Joshi	Nadiad	Amdavadi Bazar,Nadiad	1234567890	sharmanjoshi@gmail.com	0	electrician1001	we provide electrical service	Delete

< Back

Manage Category

VALORANTHome About Logout

Category ID	Category Name	Actions
1	plumbing	Delete
2	electrician	Delete
3	carpentry	Delete
4	mechanic	Delete
5	it_repair	Delete



8. Conclusion :

The functionalities are implemented in system after understanding all the system modules according to the requirements. Functionalities that are successfully implemented in the system are:

- User registration containing all the necessary validation on field
- Login
- User authentication
- Logout
- Customer can view all Service Providers Details
- Customer can Request for the Service
- Service Provider can Accept, Reject or Fulfil the Request made by Customer
- Admin can View or Delete all Service Provider & Customer

After the implementation and coding of system, comprehensive testing was Performed on the system to determine the errors and possible flaws in the system.

9. Limitations & Further Enhancement :

We are able to implement the functionality model of the “Daily Household Service”. We aim to make this product ready to be used in Only Limited Locations Currently & project supports only online Web Platform. We haven't include the Service based on Locations (like a Google Maps). We haven't allow the user & Service Provider to Upload their Profile Photo or a Shop Images.

We can add a feature of in-browser test running of the code. Further extensions involve integrating it with GitHub and integrating Machine Learning recommendation models for suggesting Service based on Past Requests. The project can be extended & support on Cross-Platform Services like on ios & Playstore. We also can implement OTP or Email based Registration also further.

10. Reference / Bibliography :

➤ Following links and websites were referred during the development of this project:

- ✓ [Django Project](#)
- ✓ [Python](#)
- ✓ [Stackoverflow](#)
- ✓ [GitHub](#)
- ✓ [w3school](#)