

CS 303-SE Lab - FINAL REPORT

URBANCLAP NITK



NITK-Urbancap Team

B Videep Kumar Reddy	191CS215
Sangamesh Kalbemullge	191CS153
Nanda Kishore KH	191CS140
K Dharmick Sai	191CS221
Prajwal J M	191CS143
Rakshit P	191CS147

Introduction

- ➔ NITK Urbanclap is intended to serve the residents of NITK(or any other college campus) with Basic Services. It also benefits the Service Providers in college campuses to smoothly connect with their customers.
- ➔ Aim of NITK Urbanclap is to allow skilled and experienced professionals to connect with NITK customers looking for specific services. Once on the platform, the customers can specify the type of service, their suitable time slot, and place a service request. The request will be then assigned to a suitable professional willing to take up the order.

Needs of users we can fulfill/outcomes.

- The Users of NITK Urbanclap will be residents of the NITK campus. The Service Providers are the Shopkeepers/(Electrician/Plumber/.. etc)/Delivery Personnels.
- The Customers can order services like Laundry, Delivery,Electricians, Cleaners,etc.
- Better Customer reach for Service Providers .
- Hassle free requests of the services you require.
- Request is just a click away in our app and we will do the rest for you :)

Project Requirements

Software Requirements

Frontend: Bootstrap, CSS, HTML

Backend: Django

Database: MySQL

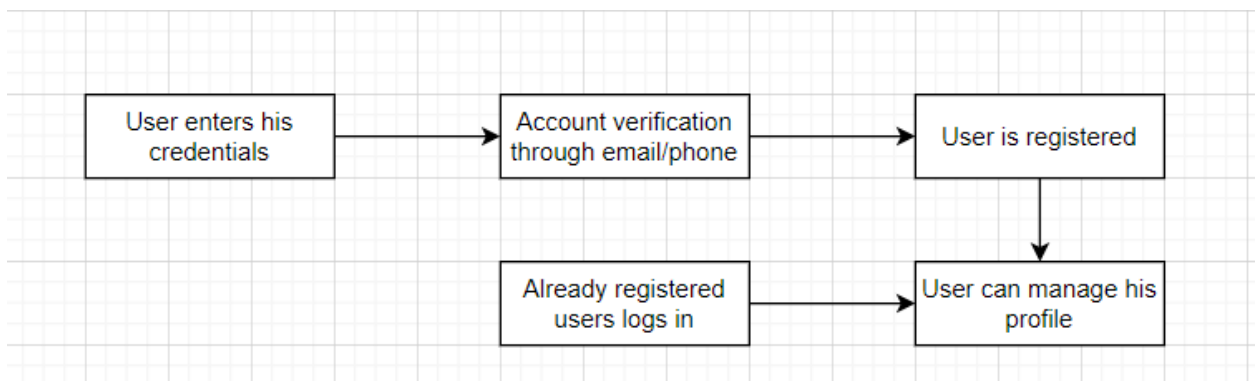
System Requirements

1. Any computer Device with Browser and internet connection.

Functional Requirements

Module 1 : User Authentication , Registration and Account Management

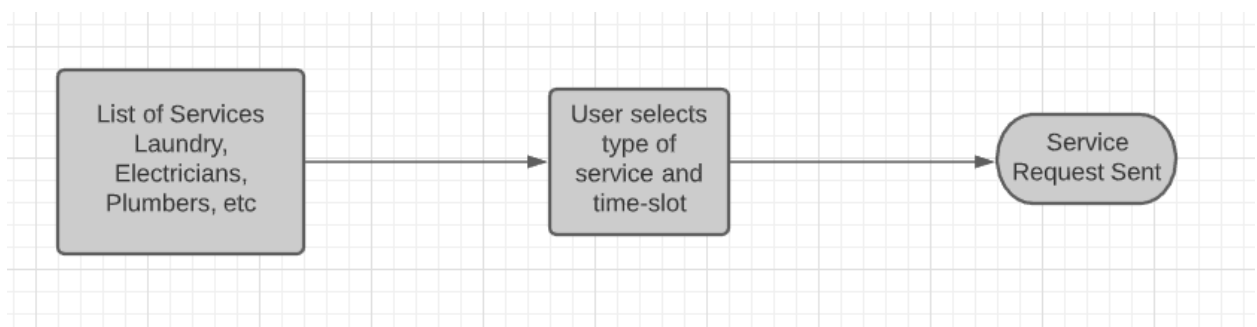
1. This includes the registration and login system , with verification through NITK mail for Students , and secondary verification for authorized Service providers only
2. This also includes the Account/Profile settings like profile pic, Password reset, Bio etc.



Module 2 : Customer and Service Provider Module

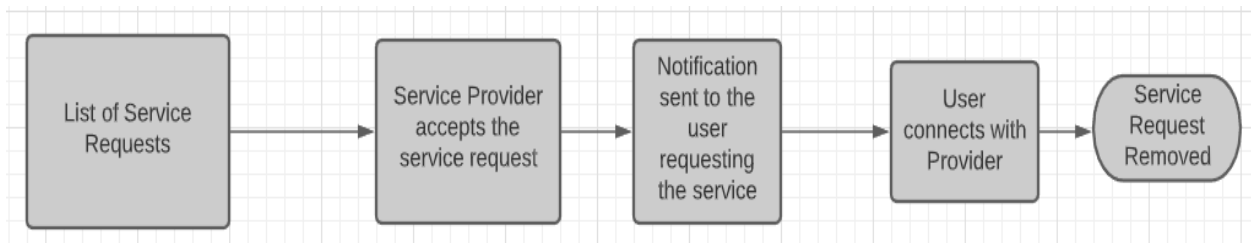
This module acts as a customer.

1. Customers can request any service
2. The Customer can also specify the time within which they order can be taken by a service provider. The Customer can also cancel the order .
3. In the Service Module Page where Service Provider can see the list of orders/services requested and accept them.
4. Flow:



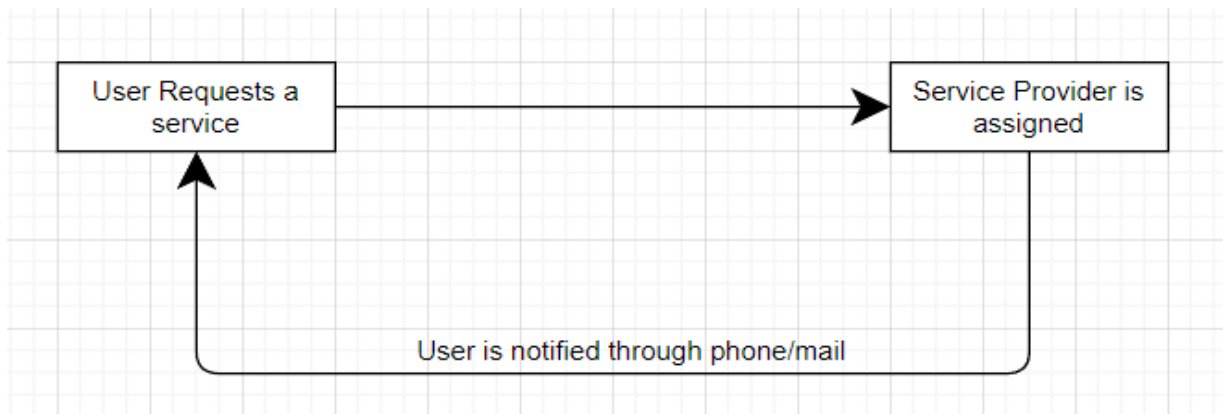
Module 3 : Order Tracking System

1. The User Can view the current stage of his order.
2. The Service Provider Regularly updates the order's stages
3. Flow:



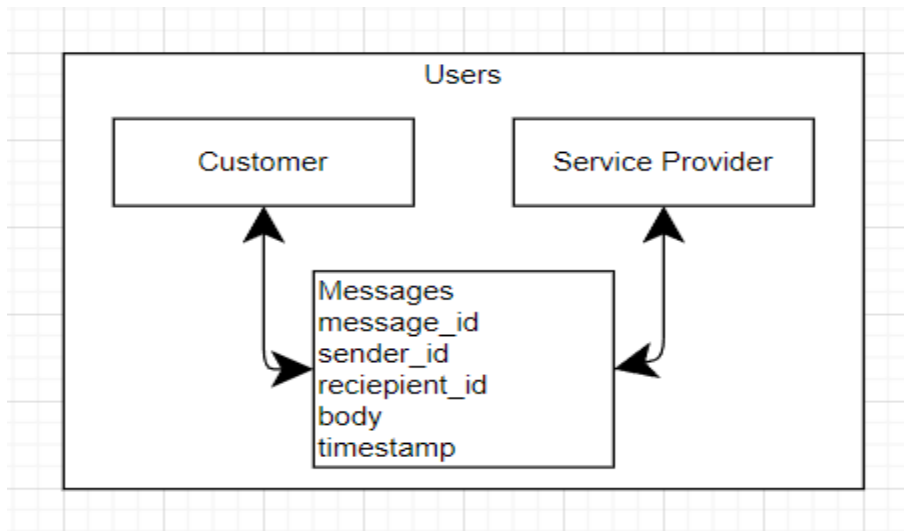
Module 4 : User Notifications: Notifies all the service providers about this request

1. Notification is sent via phone and email.
2. Users will have the option to disable notifications
3. Flow:



Module 5 : Messaging/Chatting Facilities

1. The customers and the service providers can chat with each other over the course of the order.
2. This facility connects Customer and service Provider once Order is accepted.
3. Flow:



Module 6 : Review and Feedback system

1. Customers can give reviews/feedback and rating to service providers for availed service once order is completed.
2. Service Providers can view the rating and reviews given to him to improve his work.

Module 7 : Payments

1. Integration of Paytm Payment Gateway to pay to service providers for availed service once order is completed.

Module 8 : Help and Support System

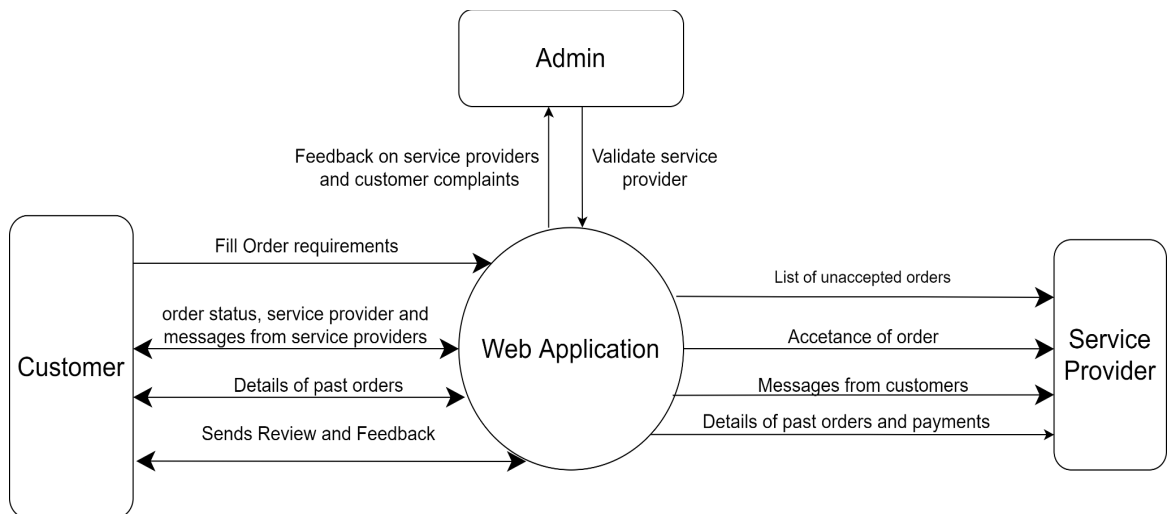
1. If there is any discrepancy when the service is being delivered they can contact the Help and Support System.
2. The Help and Support System will authenticate a service provider's account(i.e while the service provider is creating an account the Help and Support System/ team will verify if the service provider is genuine).
3. If a Service Provider is not servicing properly then the Help and Support System will suspend/ban the service provider according to the reviews and feedback.

Non Functional Requirements

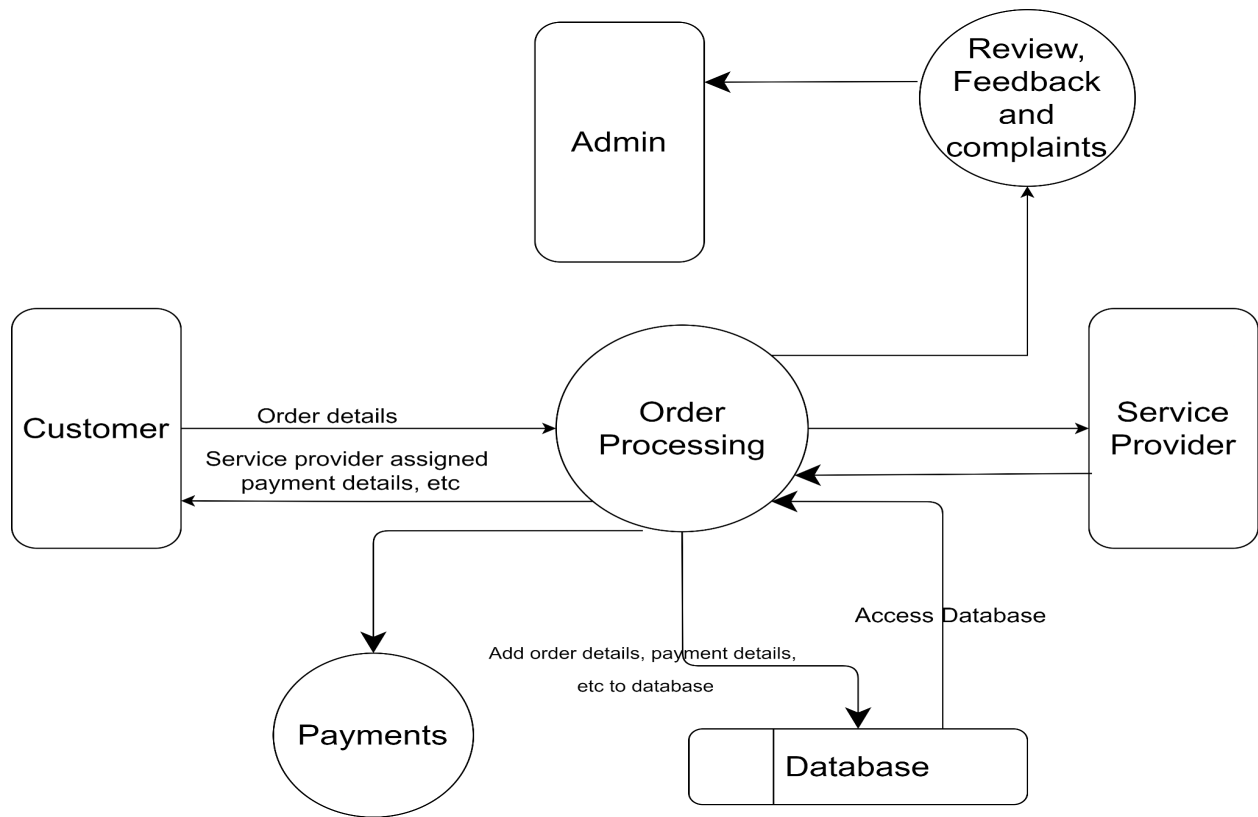
1. Performance constraints :- If we consider Reliability and Security , many functions in our application use Third Party Integrations which are very reliable and safe. We've also used Django Locks in order to avoid race conditions.
2. Operating constraints : - These include physical constraints (size, weight), and our application does not depend on any physical constraint in particular. And is also independent of any personnel availability.
3. Interface Constraints : - Interface with its environment, users, and each and every click in our application is to the point on what it exactly does, and other systems.
4. Economic constraints :- All our third party integrations and databases used are either free to use or open source , so our application runs with almost no cost required.
5. Lifecycle requirements :- For the number of users that we estimated , our application is maintainable without much of a hassle, and coming to portability there's no hindrance to that as well.

Data Flow Diagrams

Data Flow Diagram- Level 0

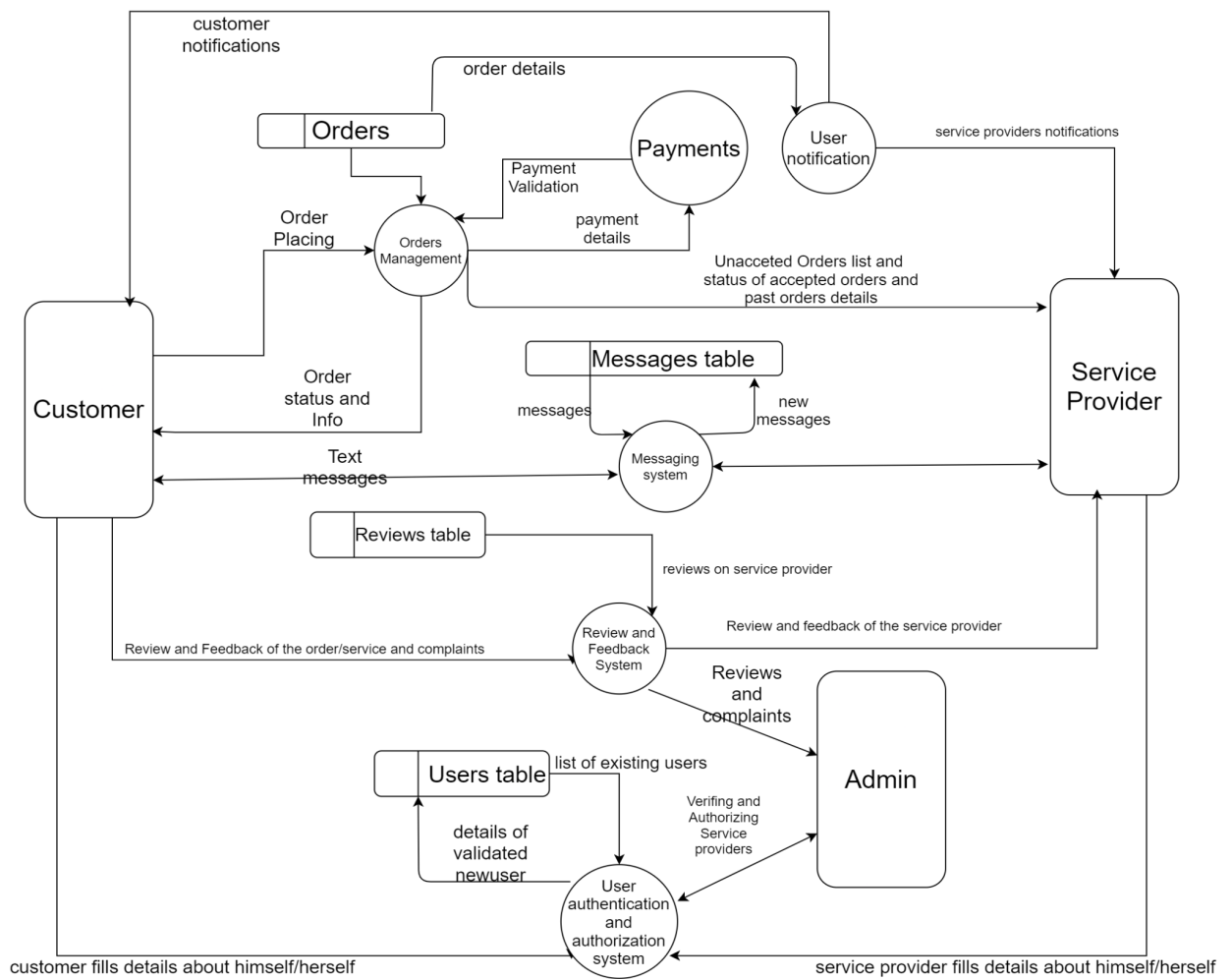


Data Flow Diagram- Level 1



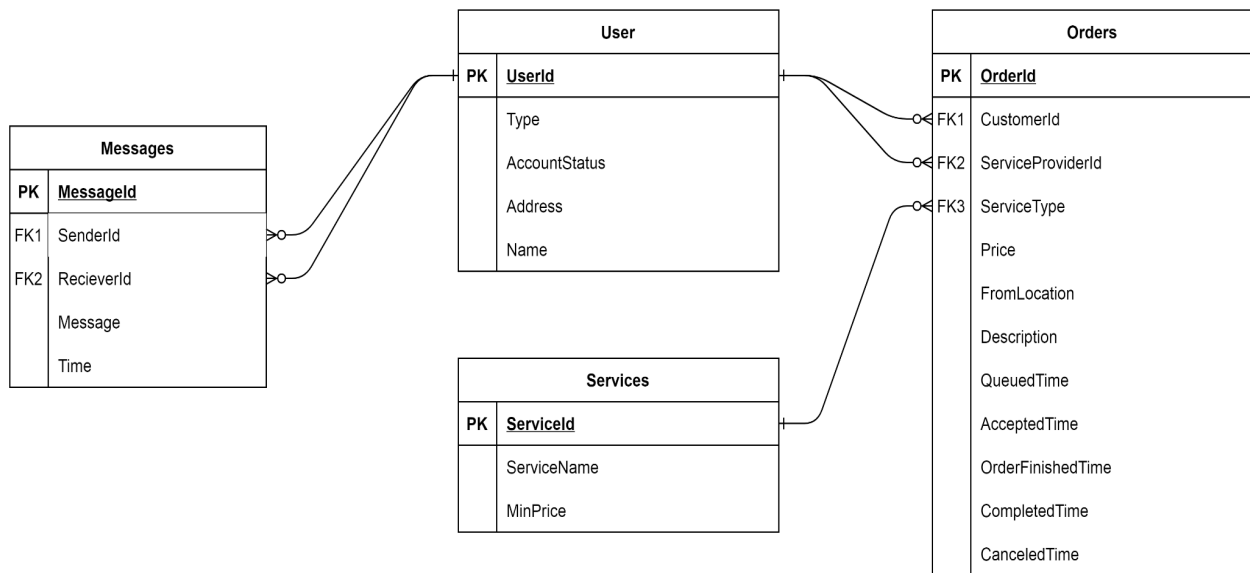
(P.T.O)

Data Flow Diagram Level 2



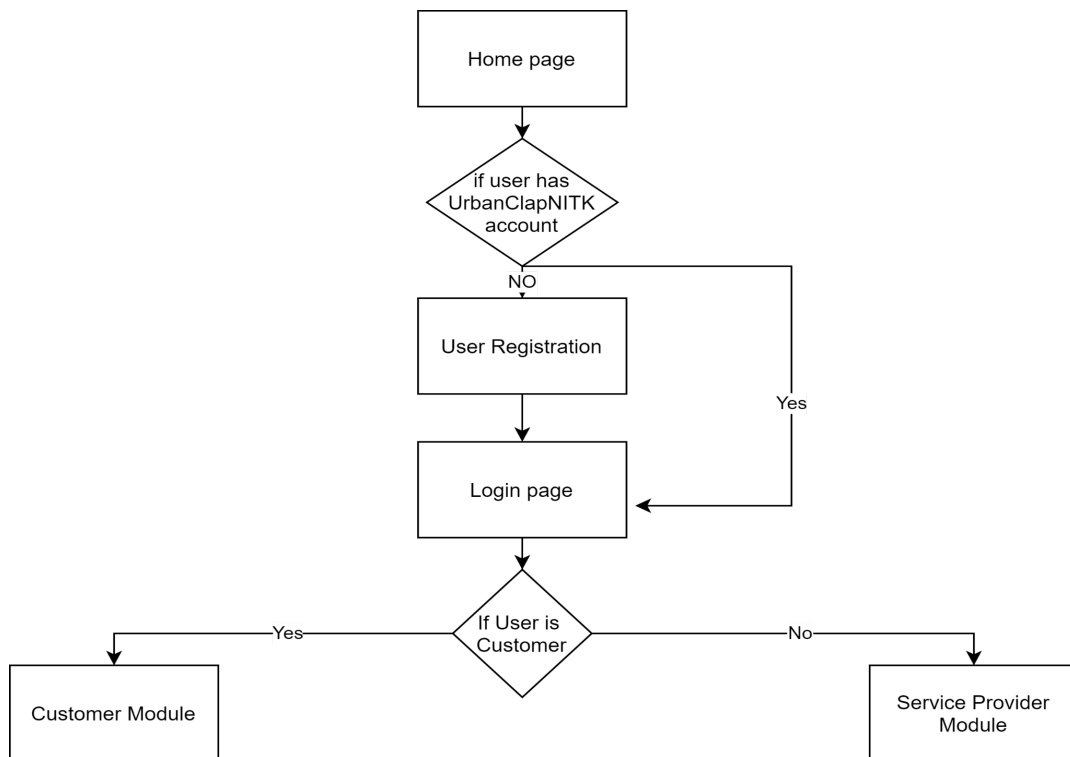
(P.T.O)

Database Schema

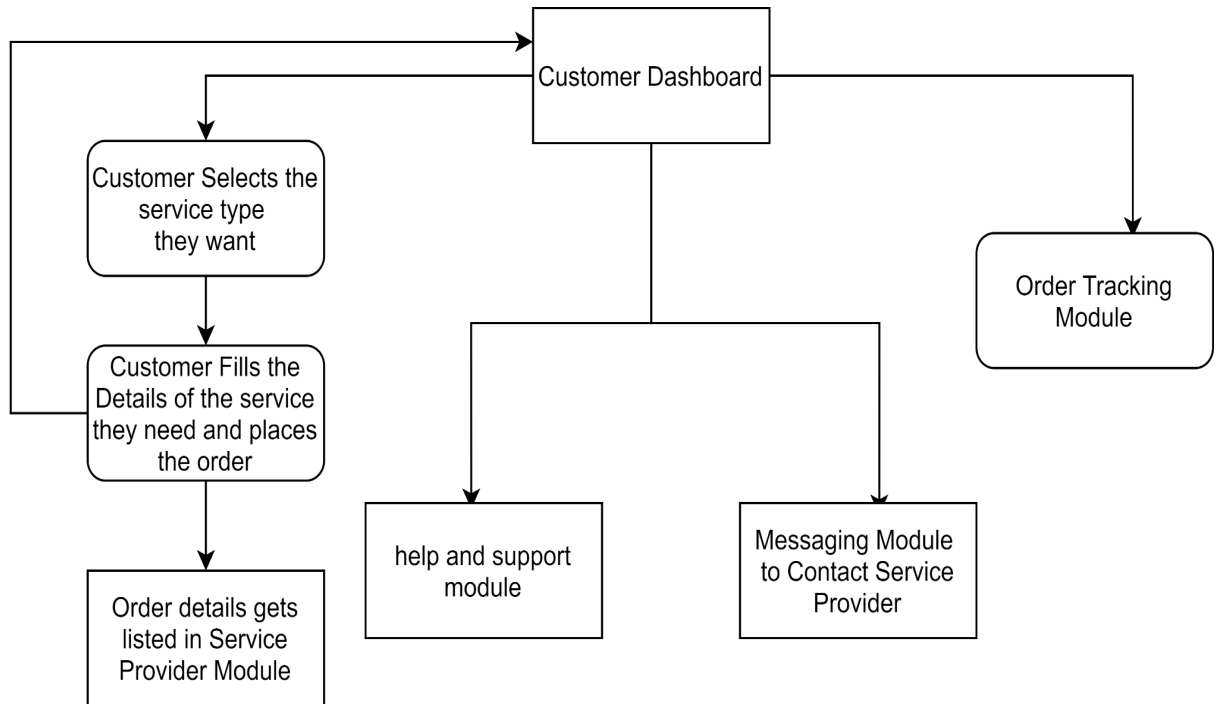


Activity Flow Diagrams

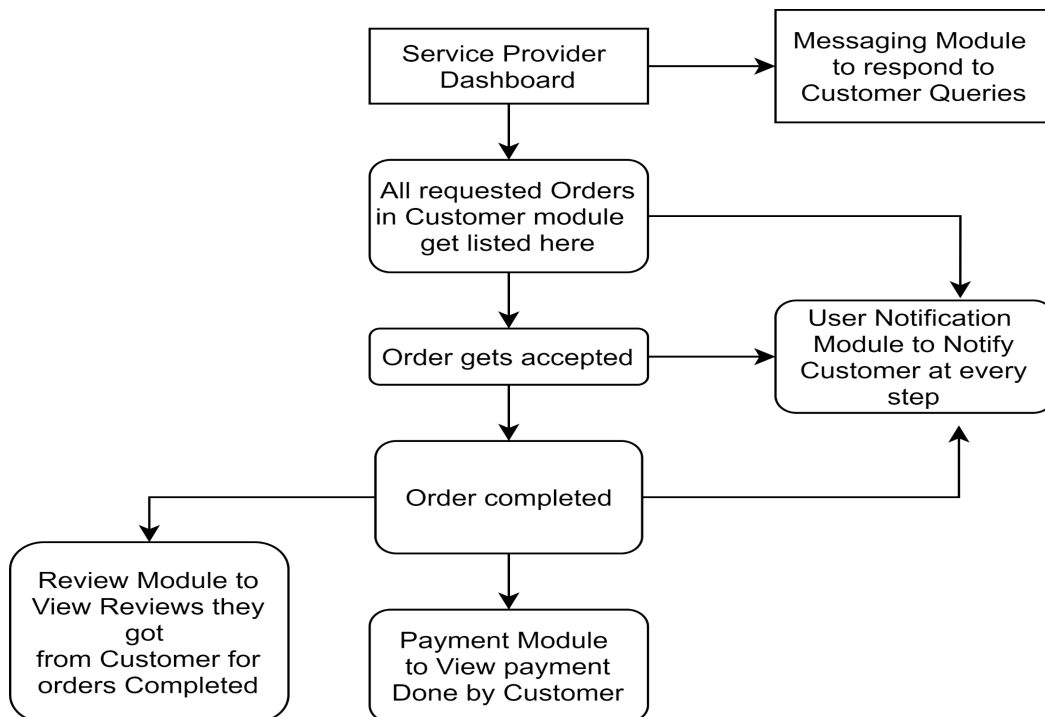
- User Authentication , Registration and Account Management Activity Flow Diagram



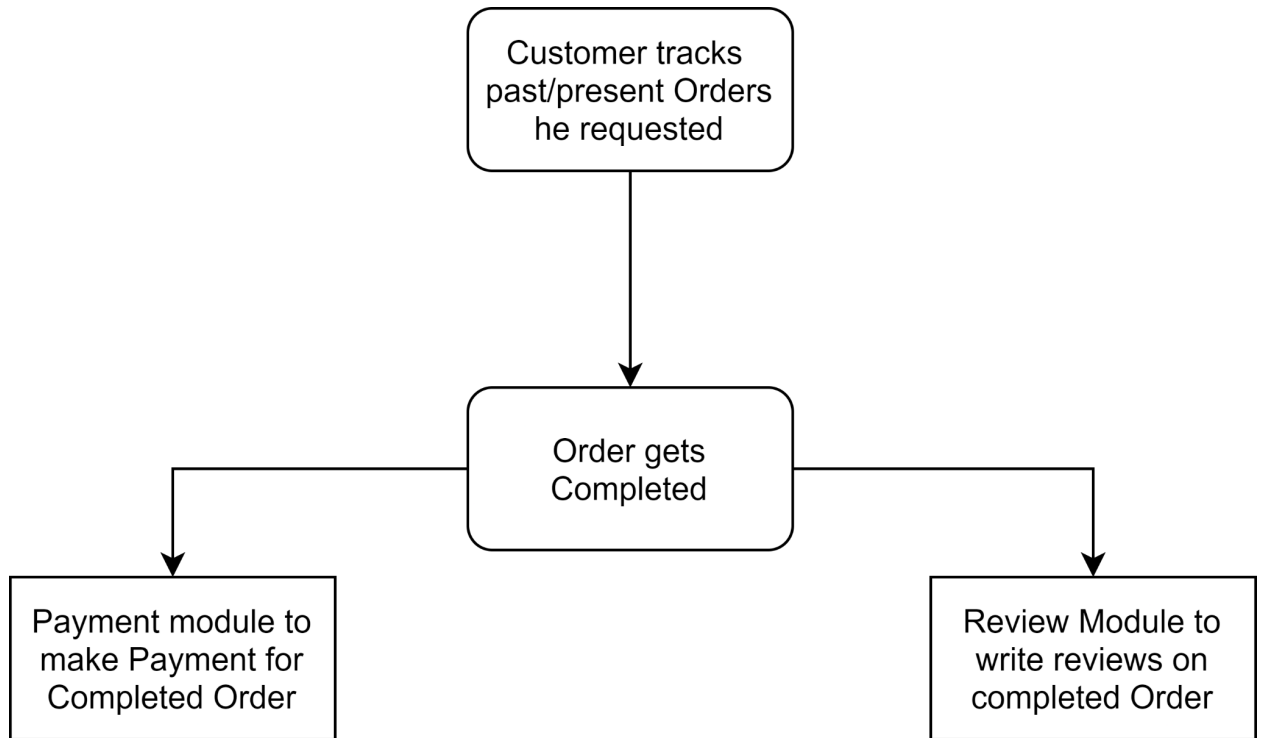
- Customer Module Activity Flow Diagram



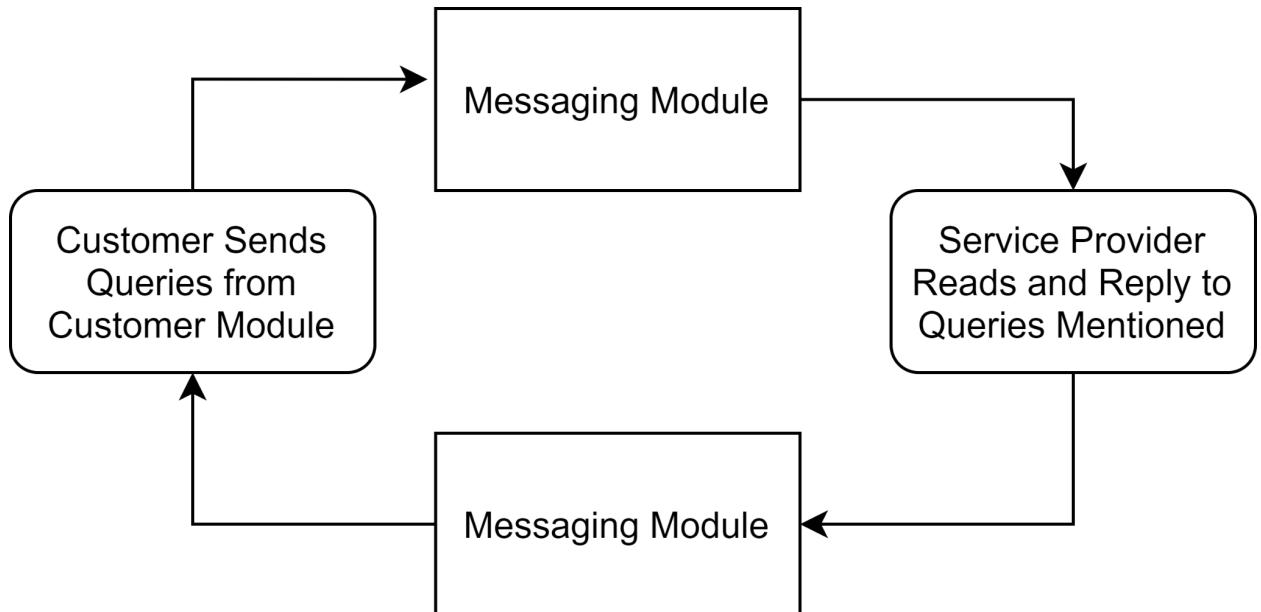
- Service Provider Module Activity Flow Diagram



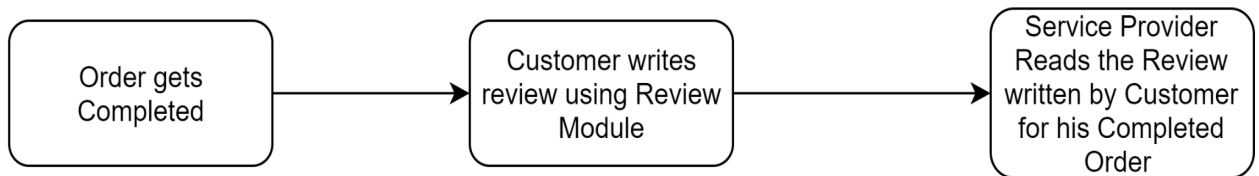
- Order Tracking System Activity Flow Diagram



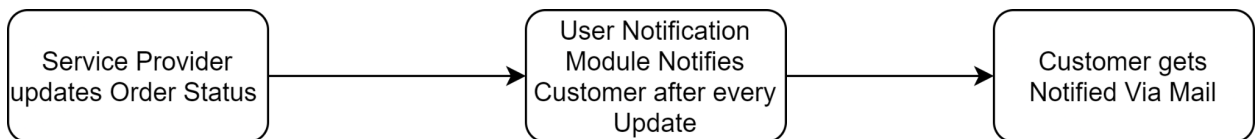
- Messaging Facility Activity Flow Diagram



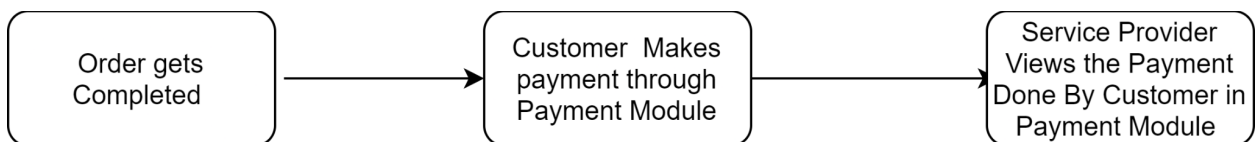
- Review Module Activity Flow Diagram



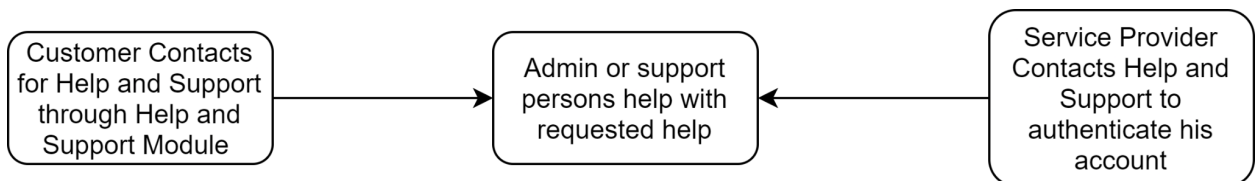
- User Notifications Activity Flow Diagram



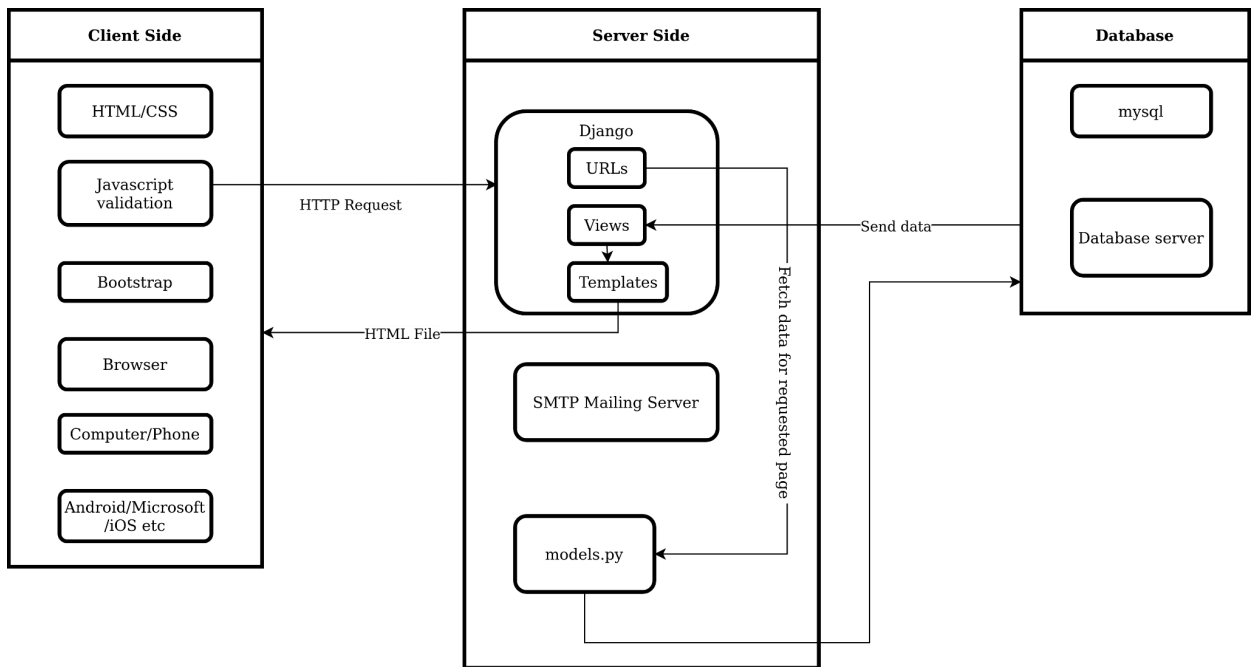
- Payments System Activity Flow Diagram



- Help and Support System Activity Flow Diagram



System Architecture



Testing report

1. Bugs found and fixed:-

a. Added Login Required decorator -

While testing we found that if we click on the 'order now' button on the homepage to make a new order without being logged in it would show an error. Similar problems happened when the user tried to access other urls of the website. This error was because we hadn't added the code on what to return when asked for a url without the user being logged into the website.

We fixed this bug by adding a 'login required' decorator to each of the functions that handled the respective urls. This decorator basically redirects the user to the login page when he tries to access the url without being logged in.

b. Django messages display -

Django messages are basically used to display success or error messages of processes like login, sign up, payment etc. While testing we found that there was no message being delivered when the customer made a successful payment on an order there was no success message being displayed on the top.

This error was because we hadn't put the django messages block on frontend. Now this error has been fixed. Here is an output screenshot after the bug was fixed.

c. Unauthorized accessing of some urls -

There is a url to handle every action in the order tracking system like accept order, cancel order, finish order etc. While testing we found that users could authorize these urls to change the order status of the order in an unauthorized manner. Like for example once the payment was done the service provider could manually type the url to accept the order and make the order back to accepted state even if the payment is done for that order. This unauthorized use of some urls have been prevented now.

d. Updating reviews -

Initially there was this bug where once the review was made there was no option to change it. Now there is an option to change the review also or re-review an order service. Writing a feedback comment was a must initially. Now that has been fixed.

Initially reviews weren't sorted for each service provider now the reviews are sorted according to the services and the service provider.

2. Statistics of testing:-

The output shows all urls concerning the web app have passed tests and no issues raised or silenced. The coverage reports of ran tests are:

Module	state			
	men ts	mis sing	exclu ded	cover age
UrbanCompanyNITK__init__.py	2	0	0	100%
UrbanCompanyNITK\asgi.py	4	4	0	0%

UrbanCompanyNITK\settings.py	40	0	0	100%
UrbanCompanyNITK\urls.py	7	1	0	86%
UrbanCompanyNITK\wsgi.py	4	4	0	0%
authentication__init__.py	0	0	0	100%
authentication\admin.py	3	0	0	100%
authentication\apps.py	4	4	0	0%
authentication\forms.py	17	0	0	100%
authentication\migrations\0001_initial.py	7	0	0	100%
authentication\migrations\0002_auto_20211004_1609.py	4	0	0	100%
authentication\migrations\0002_profile_bio.py	4	0	0	100%
authentication\migrations\0003_merge_0002_auto_20211004_1609_0002_profile_bio.py	4	0	0	100%
authentication\migrations\0003_merge_20211005_0621.py	4	0	0	100%
authentication\migrations\0003_merge_20211005_0713.py	4	0	0	100%
authentication\migrations\0003_merge_20211005_1154.py	4	0	0	100%
authentication\migrations\0004_auto_20211008_1215.py	4	0	0	100%
authentication\migrations\0005_merge_20211009_0652.py	4	0	0	100%
authentication\migrations\0006_merge_20211009_1215.py	4	0	0	100%
authentication\migrations\0007_auto_20211011_0749.py	4	0	0	100%
authentication\migrations\0008_auto_20211011_1320.py	4	0	0	100%
authentication\migrations\0008_merge_20211011_1122.py	4	0	0	100%
authentication\migrations\0009_auto_20211011_1933.py	4	0	0	100%
authentication\migrations\0010_merge_20211012_0534.py	4	0	0	100%
authentication\migrations\0011_alter_profile_id.py	4	0	0	100%
authentication\migrations\0012_auto_20211016_1252.py	4	0	0	100%
authentication\migrations__init__.py	0	0	0	100%
authentication\models.py	20	7	0	65%
authentication\tests.py	1	1	0	0%

authentication\urls.py	4	0	0	100%
authentication\utils.py	6	1	0	83%
authentication\views.py	127	94	0	26%
direct__init__.py	0	0	0	100%
direct\admin.py	1	0	0	100%
direct\apps.py	3	3	0	0%
direct\migrations\0001_initial.py	7	0	0	100%
direct\migrations__init__.py	0	0	0	100%
direct\models.py	22	10	0	55%
direct\tests.py	1	1	0	0%
direct\urls.py	3	0	0	100%
direct\views.py	74	57	0	23%
manage.py	12	2	0	83%
ucnitk__init__.py	0	0	0	100%
ucnitk\admin.py	6	0	0	100%
ucnitk\apps.py	4	4	0	0%
ucnitk\forms.py	12	0	0	100%
ucnitk\migrations\0001_initial.py	8	0	0	100%
ucnitk\migrations\0002_auto_20211020_0023.py	4	0	0	100%
ucnitk\migrations__init__.py	0	0	0	100%
ucnitk\models.py	46	2	0	96%
ucnitk\tests.py	50	0	0	100%
ucnitk\urls.py	5	0	0	100%
ucnitk\views.py	179	131	0	27%
Total	747	326	0	56%

Here statements are total no of statements in that file and missing is total no of statements missed and coverage is total covered and excluded are classes of code which are to be excluded.

Maintenance Report

Security Issues

- **Razorpay Account Credentials**
 1. Initially Razorpay Account Id was visible in the codebase which was a big security bug and could be easily misused by users.
 2. These were later moved to a .env file which is gitignored in the codebase, thus protecting the credentials. Also some credentials were there in settings.py which weren't safe but now we have put it in .env file
- **Password Strength**
 1. Any user password was being accepted initially.
 2. Now, the strength of the password is checked by looking at various criterias like the length of the password, presence of digits, alphabets, etc and the user is prompted to make a strong password.
 3. Even the website administrator credentials were strengthened to prevent hacking of the website.

Register for a free account

Password is not alphanumeric

Nandha

kdharmicksal.191cs221@nitk.edu.in

Password

SHOW

Customer Delivery

Register

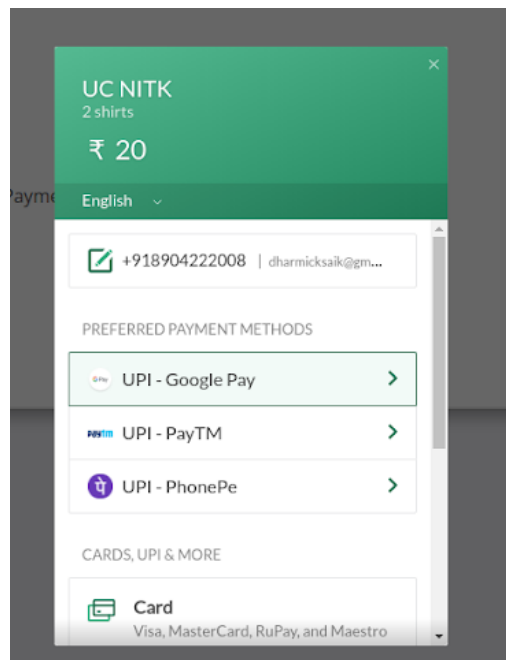
- **Strengthening Database Transactions**

1. Triggers in the front-end like clicking on an “Order” button executes a database transaction. These triggers could also be executed if a user manually goes to a specific url or manipulates components in the developer tools.
2. For example, a service provider could try to accept an order that has been already accepted by another service provider, if they know the URL route at which the backend redirects when an order gets accepted.
3. To prevent such illegal transactions to the database, before any transaction is done, different checks are carried out to verify the details of the transaction.
4. In the example given above, the checks would reject the service provider’s illegal request as that particular order had been already accepted.

Maintenance Bugs

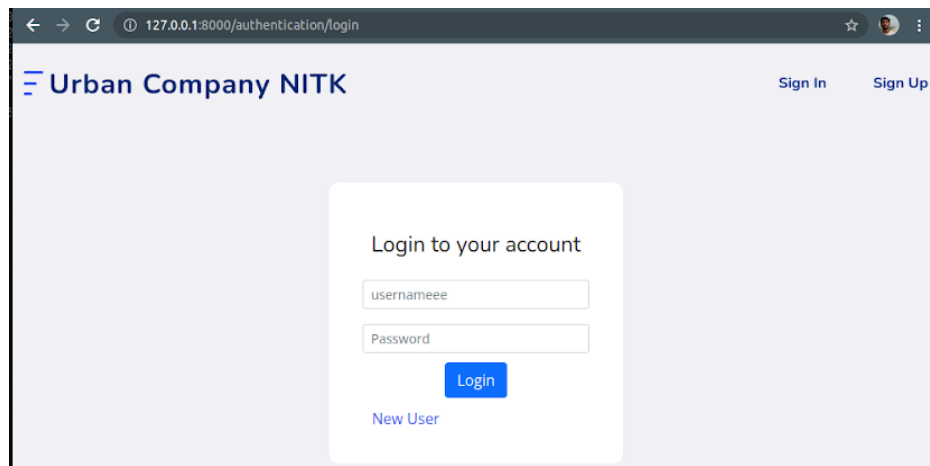
- **Live Payments**

1. In the development phase, we could only perform demo payments using Razorpay.
2. In the maintenance phase, we generated api keys in Razorpay to support live payment



- **Functionalities in User Authentication Page**

1. The Login page and the Sign up page were standalone and could only be used for the specific purpose of that page.
2. In the maintenance phase, we added different redirect options in these pages, like to the homepage of the website, to the sign up page when the user is trying to login, etc.



- **Improving the UI of the Messaging functionality**

1. Initially both the User's message and the sender's message were on the same side of the screen which made it hard for readability, now the user's messages are on the right side of screen and the other person's message is on the left side of the screen, giving better clarity.



- **Fixing version of packages while deploying**

1. The initial version of deployment did not take into account that the packages used in the application would be modified later. This could potentially give rise to bugs in the future.
2. To prevent such bugs in the maintenance phase, we added requirements.txt and maintained a steady version of all the packages used. This would also be followed by updating the packages as and when required.

Releases

- **Version 1.0**

- Contained user-authentication, user profile and the order tracking modules forming the core of the web application.

- **Version 2.0**

- Notifications support was added to notify the users during a change in the order status.
- Messaging facilities between users were added.
- Demo payments were supported using Razorpay

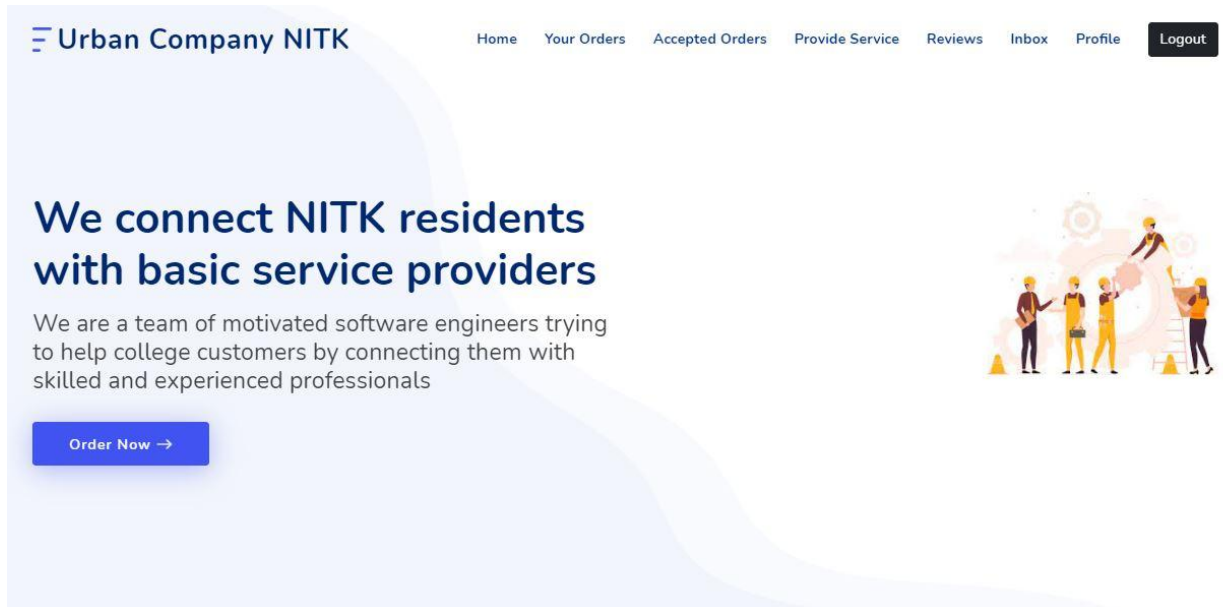
- **Version 3.0**

- Functionality to give ratings and reviews to service providers was added.
- Live payments were integrated thus enabling users to actually make their payments for orders.
- A Help and Support System was set up for customers and service providers
- Fixed Security issues in database transactions and user credentials
- Tests were added to check the correctness of the application

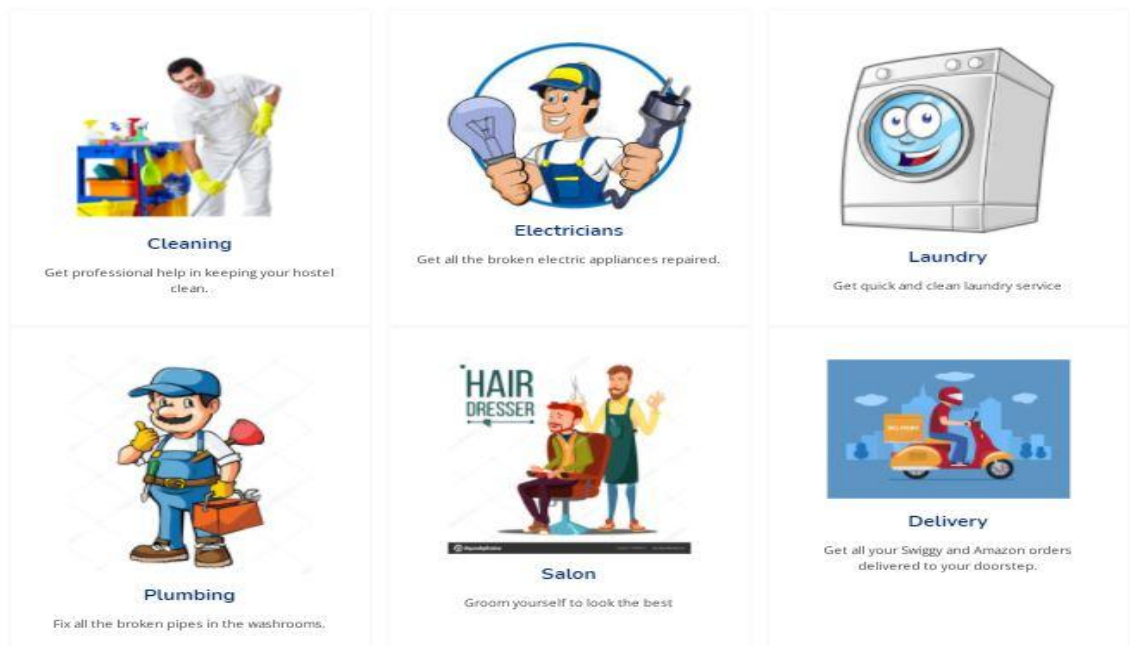
(P.T.O)

Screenshots of final project

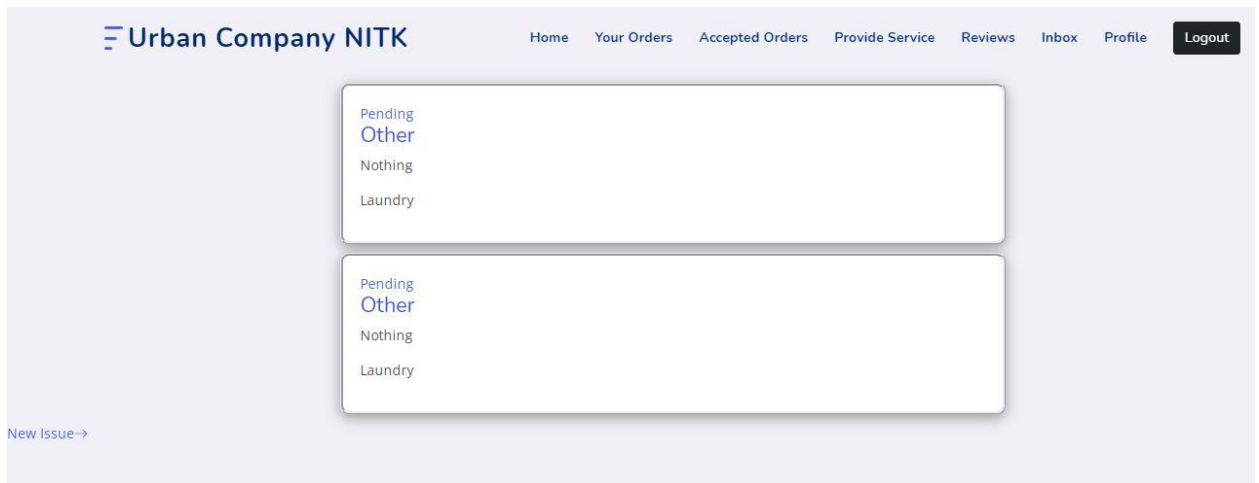
- Home page (Landing Page)



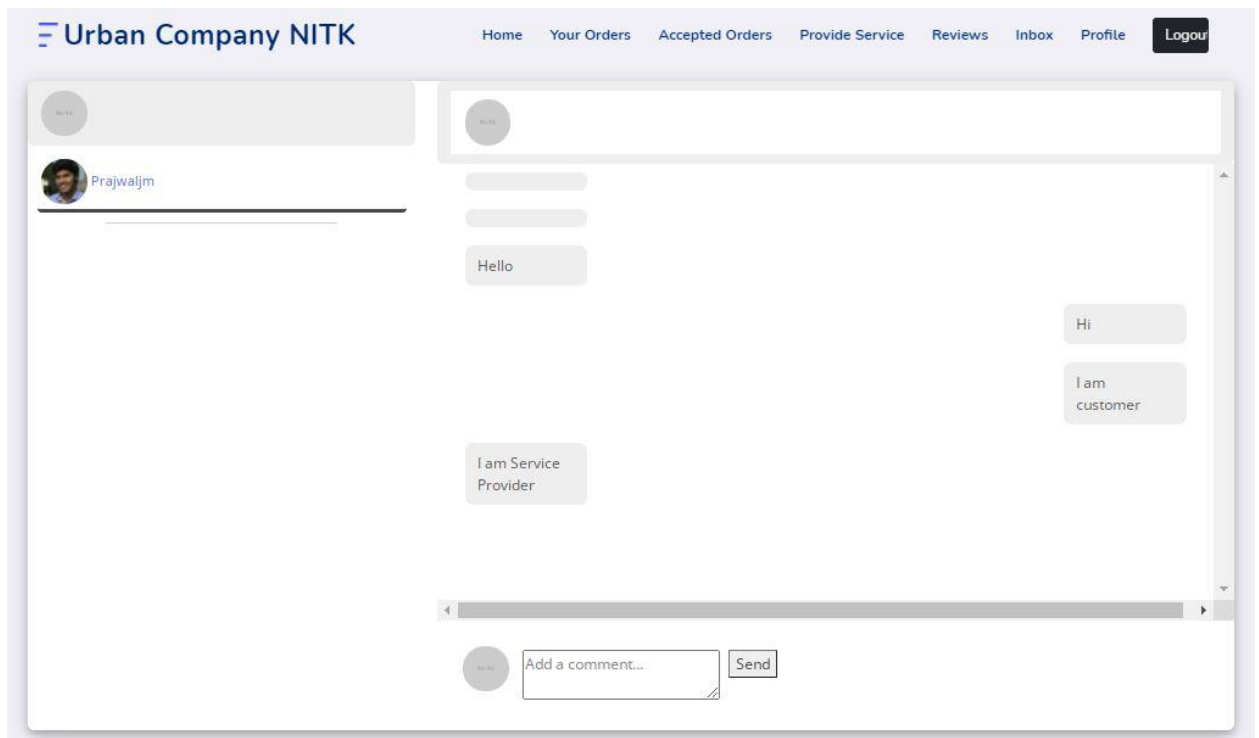
Services we provide



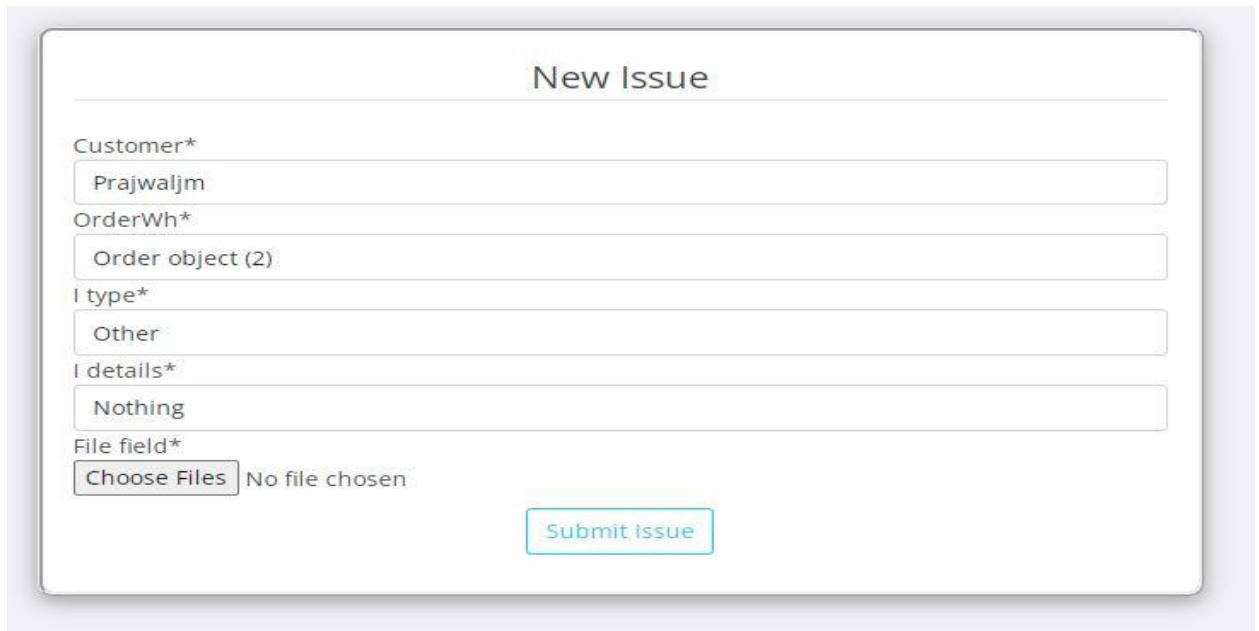
- **Issue list**



- **Messaging (Inbox)**



- **Help and support issue form**



The 'New Issue' form is a white rectangular box with a light gray border. It contains several input fields and a submit button. The fields are labeled 'Customer*', 'OrderWh*', 'I type*', 'I details*', and 'File field*'. The 'Customer*' field contains the text 'Prajwaljm'. The 'OrderWh*' field contains the text 'Order object (2)'. The 'I type*' field contains the text 'Other'. The 'I details*' field contains the text 'Nothing'. The 'File field*' field contains a 'Choose Files' button and the text 'No file chosen'. A 'Submit Issue' button is located at the bottom right of the form.

New Issue

Customer*

Prajwaljm

OrderWh*

Order object (2)

I type*

Other

I details*

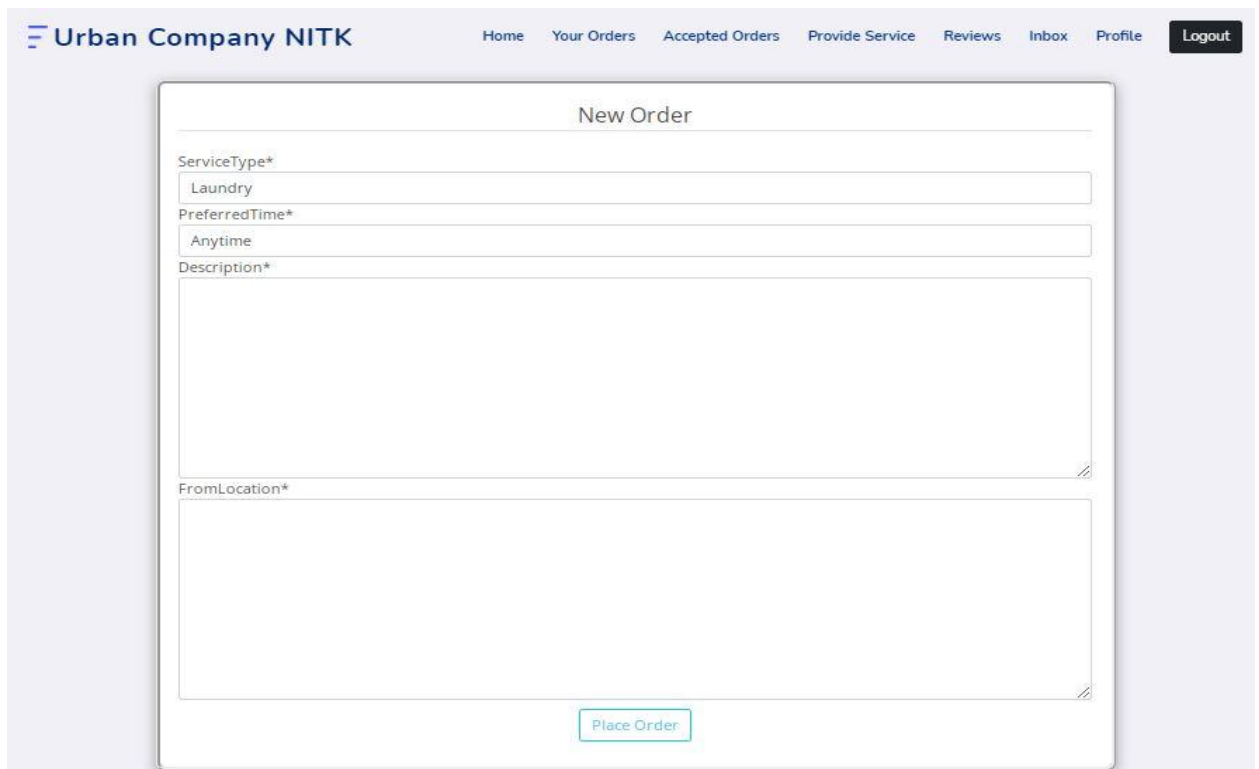
Nothing

File field*

Choose Files No file chosen

Submit Issue

- **Order form**



The 'New Order' form is a white rectangular box with a light gray border. It is part of a larger web application interface. The header of the application shows 'Urban Company NITK' on the left and navigation links 'Home', 'Your Orders', 'Accepted Orders', 'Provide Service', 'Reviews', 'Inbox', 'Profile', and a 'Logout' button on the right. The form itself contains input fields for 'ServiceType*', 'PreferredTime*', 'Description*', and 'FromLocation*'. The 'ServiceType*' field contains the text 'Laundry'. The 'PreferredTime*' field contains the text 'Anytime'. The 'Description*' field is a large text area. The 'FromLocation*' field is a large text area. A 'Place Order' button is located at the bottom right of the form.

Urban Company NITK

Home Your Orders Accepted Orders Provide Service Reviews Inbox Profile Logout

New Order

ServiceType*

Laundry

PreferredTime*

Anytime

Description*

FromLocation*

Place Order

- **Order details**

Prajwaljm
Laundry
Order 6

Description : 5 pants, 7 shirts, 1 towel,

Preferred Time : 2PM-5PM

The Order's status is Not Accepted

[help and support](#) [Delete Order](#)

Prajwaljm
Laundry
Order 6

Description : 5 pants, 7 shirts, 1 towel,

Preferred Time : 2PM-5PM

The Order's status is Pending Payment

[help and support](#)

Quoted price : INR 15

[Pay Now](#)

Prajwaljm
Laundry
Order 6

Description : 5 pants, 7 shirts, 1 towel,

Preferred Time : 2PM-5PM

The Order's status is Not Accepted

[Accept Order](#)

Prajwaljm

Laundry

Order 6

Description : 5 pants, 7 shirts, 1 towel,

Preferred Time : 2PM-5PM

The Order's status is Accepted

Cancel Order

Quote price for Service :

Finish Order

Prajwaljm

Laundry

Order 6


Description : 5 pants, 7 shirts, 1 towel,

Preferred Time : 2PM-5PM

The Order's status is Pending Payment

Confirm payment by Customer

- **User profile**



Prajwaljm
prajwaljm.191cs143@nitk.edu.in
3rd year student at NITK CSE

Profile Info

Username*

Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Image*

Currently: [profile_pics/Prajwal_J_M-profile-pic_mmOH1X2.jpg](#)

Change: No file chosen

(P.T.O)

- **Razorpay Payment gateway**

The image shows a Razorpay payment modal overlaid on a web page. The modal is titled 'UC NITK' and lists the items '5 pants, 7 shirts, 1 towel' with a total price of '₹ 15'. It includes a language dropdown set to 'English', a country dropdown set to '+91', a phone number field, and an email field. A security notice at the bottom states 'This payment is secured by Razorpay.' and a green 'PROCEED' button is at the bottom.

Prajwaljm
Laundry
Order 6
Description : 5 pants, 7 shirts, 1 towel
Preferred Time : 2PM-5PM
The Order's status is Pending
[help and support](#)
Quoted price : INR 15
[Pay Now](#)

UC NITK
5 pants, 7 shirts, 1 towel,
₹ 15

English ▾

Country
+91 ▾ Phone

Email

This payment is secured by Razorpay.

PROCEED

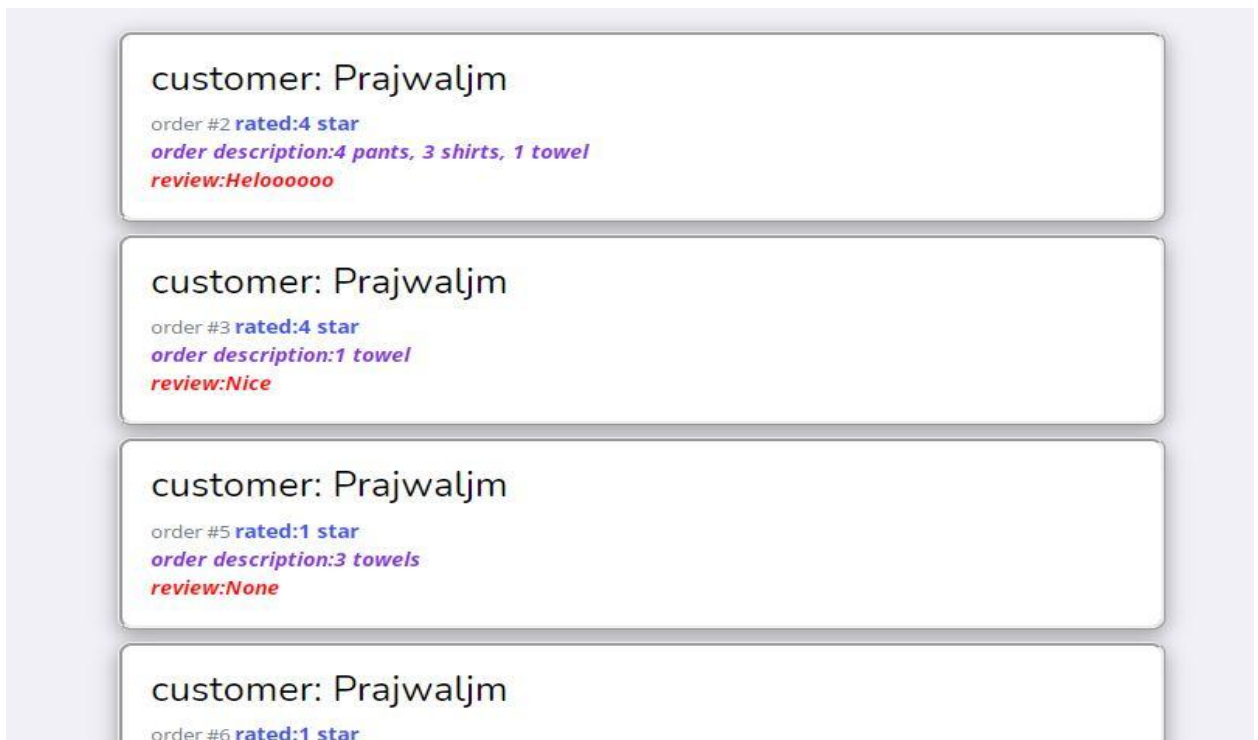
(P.T.O)

- **Feedback Form**



A feedback form titled "review order". It contains a text input field with the placeholder "rate(1 to 5)", a larger text area with the placeholder "Enter Feedback", and a "submit" button at the bottom left.

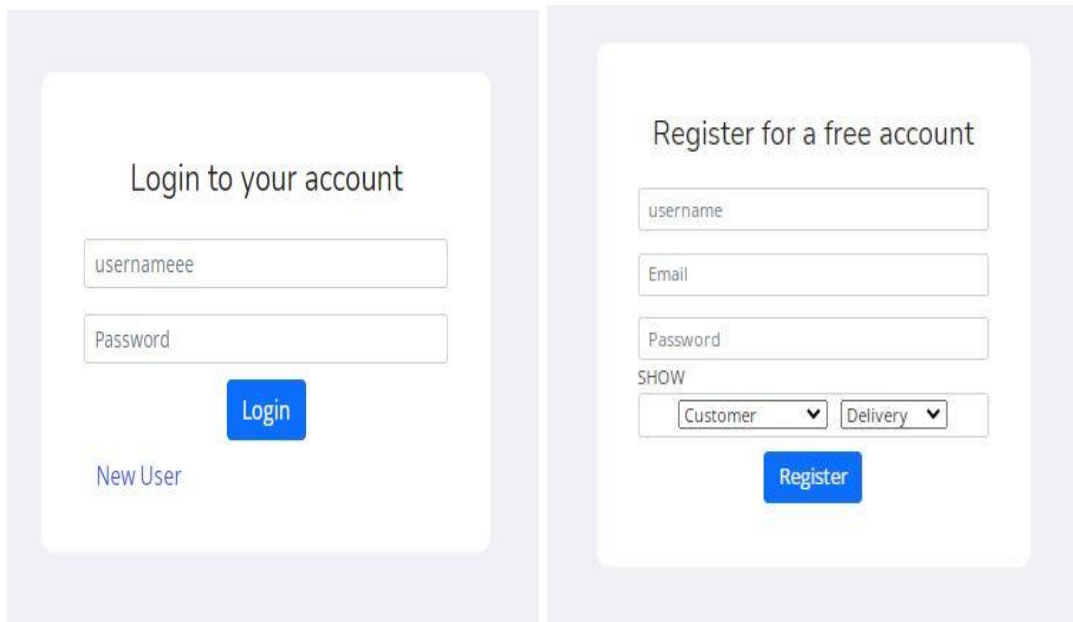
- **Customer Reviews: All customer reviews to service providers can be seen here**



A list of four customer reviews for a service provider named Prajwaljm. Each review entry includes the customer name, order number, rating, order description, and the review text.

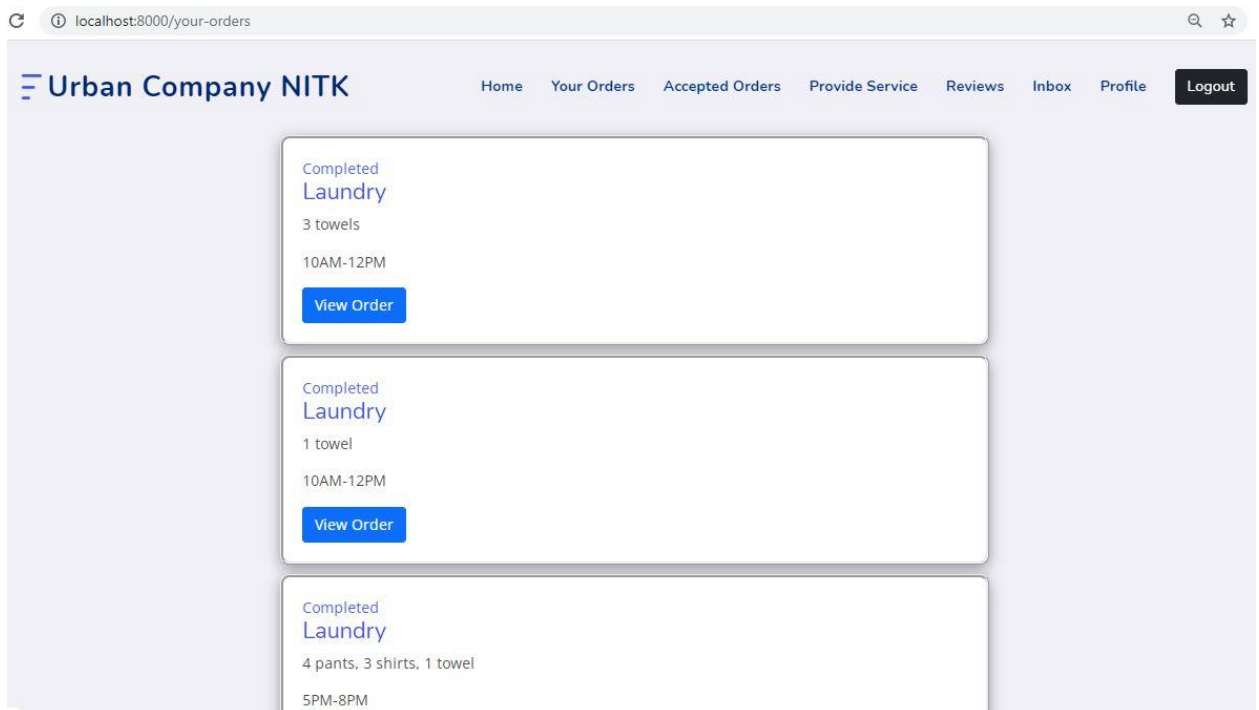
customer	order #	rated	order description	review
Prajwaljm	#2	4 star	4 pants, 3 shirts, 1 towel	Heloooooooo
Prajwaljm	#3	4 star	1 towel	Nice
Prajwaljm	#5	1 star	3 towels	None
Prajwaljm	#6	1 star		

- **Login and Register pages**



The image displays two side-by-side web forms. The left form is titled 'Login to your account' and contains input fields for 'usernameeee' and 'Password', a blue 'Login' button, and a link for 'New User'. The right form is titled 'Register for a free account' and contains input fields for 'username', 'Email', and 'Password'. Below these is a 'SHOW' section with two dropdown menus, one set to 'Customer' and the other to 'Delivery', followed by a blue 'Register' button.

- **Your Orders page: All your orders will be listed here**



This is a screenshot of a web browser showing the 'Your Orders' page for 'Urban Company NITK'. The browser's address bar shows 'localhost:8000/your-orders'. The page has a navigation bar with links: Home, Your Orders, Accepted Orders, Provide Service, Reviews, Inbox, Profile, and a Logout button. The main content area lists three completed laundry orders. Each order card shows the status 'Completed', the item 'Laundry', the quantity, the time slot, and a 'View Order' button.

Order Status	Item	Quantity	Time Slot	Action
Completed	Laundry	3 towels	10AM-12PM	View Order
Completed	Laundry	1 towel	10AM-12PM	View Order
Completed	Laundry	4 pants, 3 shirts, 1 towel	5PM-8PM	

THANK YOU