MINI PROJECT

(2020-21)

“Drop Ship”

Project Report

Institute of Engineering & Technology

Submitted By -

Abhishek Jadon(191500026)

Jay Thakur (191500368)

Dushyant Verma(191500282)

Aryan Gupta(191500157)

Under the Supervision Of

Mr. Abhishek Kr. Tiwari

Technical Trainer

Department of Computer Engineering & Applications

Department of Computer Engineering and Applications

GLA University, 17 km. Stone NH#2, Mathura-Delhi Road,

Chaumuha, Mathura – 281406 U.P (India)

Declaration

I/we hereby declare that the work which is being presented in the Bachelor of technology. Project **"Drop Ship"**, in partial fulfilment of the requirements for the award of the ***Bachelor of Technology*** in Computer Science and Engineering and submitted to the Department of Computer Engineering and Applications of GLA University, Mathura, is an authentic record of my/our own work carried under the supervision of **Mr. Abhishek Kr. Tiwari, Technical Trainer, Dept. of CEA, GLA University.**

The contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree.

Sign: *Dushyant Verma*

Name of Candidate: Dushyant Verma

University Roll No.:191500282

Sign: *Aryan Gupta*

Name of Candidate: Aryan Gupta

University Roll No.:191500157

Sign: *Jay Thakur*

Name of Candidate: Jay Thakur

University Roll No.:191500368

Sign: *Abhishek Jadon*

Name of Candidate: Abhishek Jadon

University Roll No.:191500026

Department of Computer Engineering and Applications

GLA University, 17 km. Stone NH#2, Mathura-Delhi Road,

Chaumuha, Mathura – 281406 U.P (India)

Certificate

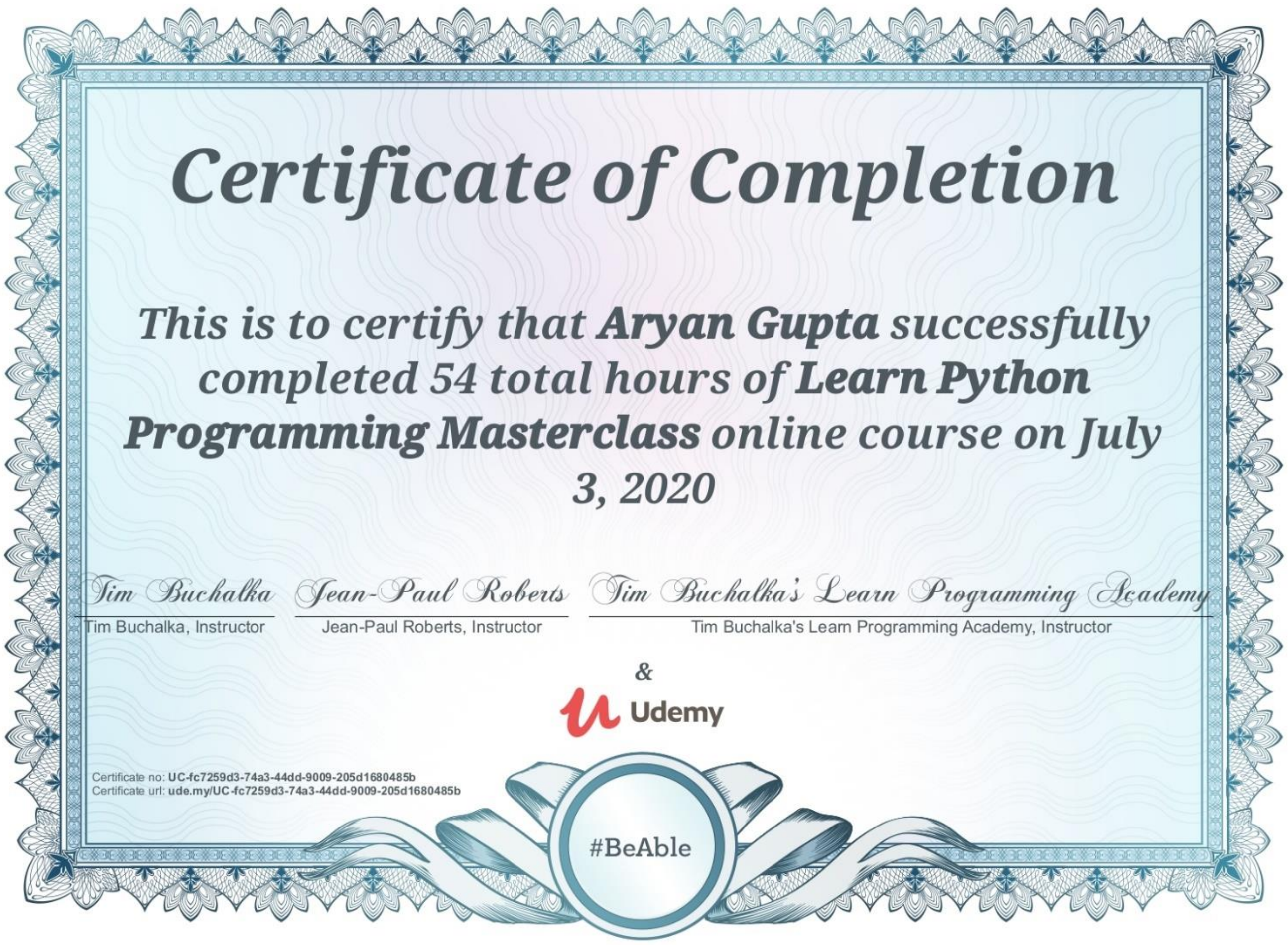
This is to certify that the project entitled "**Drop Ship**", carried out in Mini Project – I Lab, is a Bonafede work by *Dushyant Verma*, Aryan Gupta, *Jay Thakur*, *Abhishek Jadon* and is submitted in partial fulfilment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

Signature of Supervisor:

Name of Supervisor: Mr. Abhishek Tiwari

Training Certificates

- **Aryan Gupta**



Certificate of Completion

This is to certify that **Aryan Gupta** successfully completed 54 total hours of **Learn Python Programming Masterclass** online course on July 3, 2020

Tim Buchalka, Instructor

Jean-Paul Roberts, Instructor

Tim Buchalka's Learn Programming Academy, Instructor

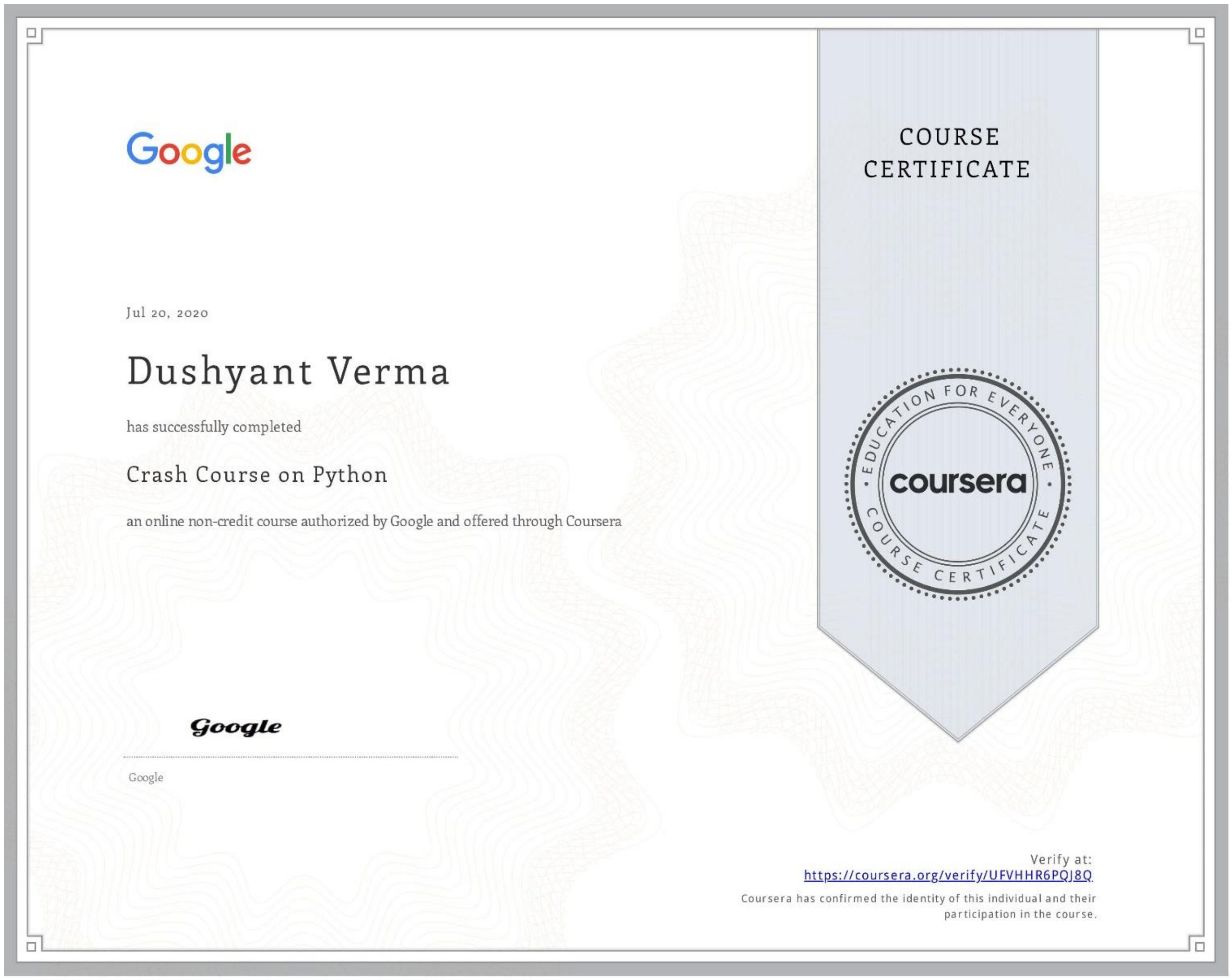
&

Udemy

Certificate no: **UC-fc7259d3-74a3-44dd-9009-205d1680485b**
Certificate url: **ude.my/UC-fc7259d3-74a3-44dd-9009-205d1680485b**

#BeAble

- Dushyant Verma


Google

COURSE CERTIFICATE

Jul 20, 2020

Dushyant Verma

has successfully completed

Crash Course on Python

an online non-credit course authorized by Google and offered through Coursera

Google

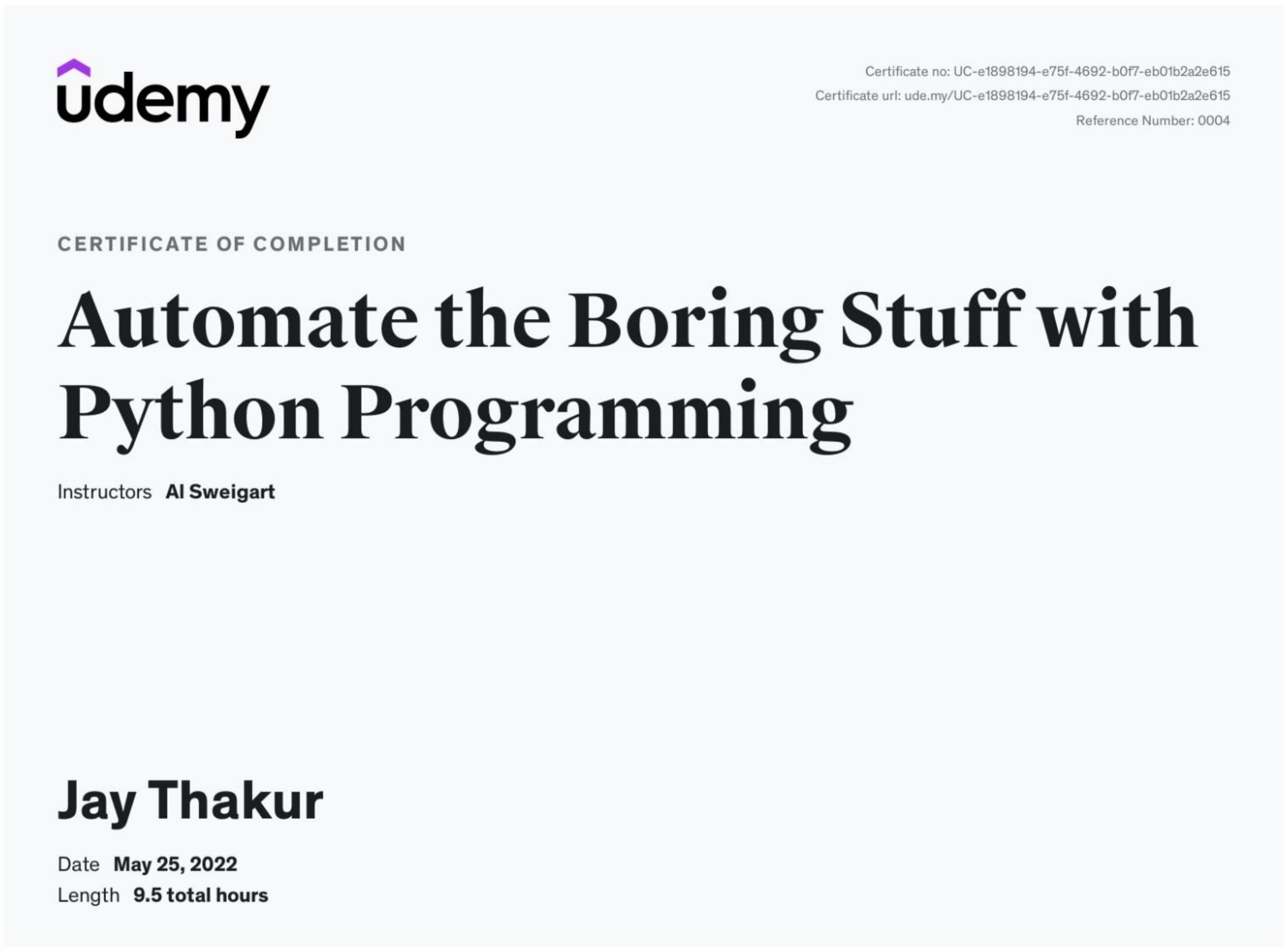
Google

EDUCATION FOR EVERYONE coursera COURSE CERTIFICATE

Verify at:
https://coursera.org/verify/UFVHHR6PQJ8Q

Coursera has confirmed the identity of this individual and their participation in the course.

- Jay Thakur



udemy

Certificate no: UC-e1898194-e75f-4692-b0f7-eb01b2a2e615
Certificate url: ude.my/UC-e1898194-e75f-4692-b0f7-eb01b2a2e615
Reference Number: 0004

CERTIFICATE OF COMPLETION

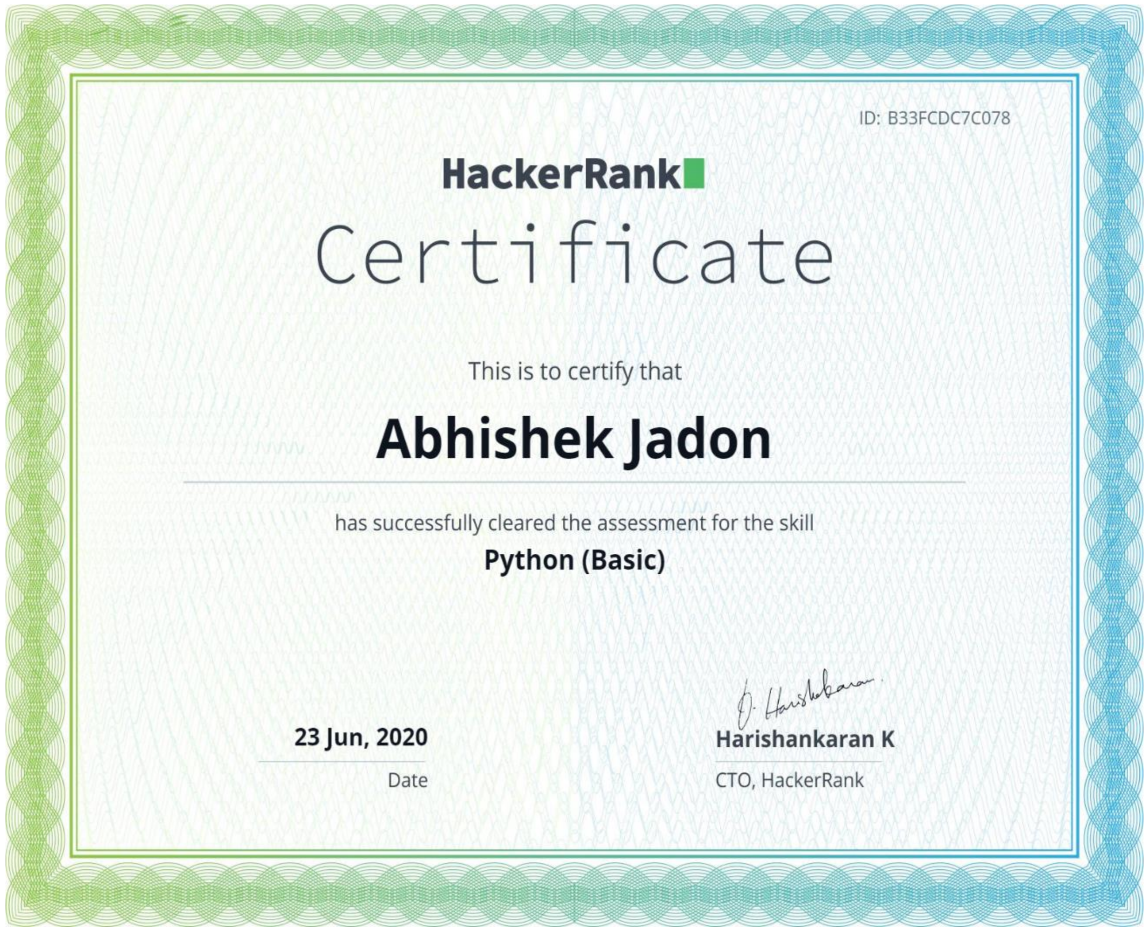
Automate the Boring Stuff with Python Programming

Instructors **Al Sweigart**

Jay Thakur

Date **May 25, 2022**
Length **9.5 total hours**

- **Abhishek Jadon**



ID: B33FCDC7C078

HackerRank

Certificate

This is to certify that

Abhishek Jadon

has successfully cleared the assessment for the skill

Python (Basic)

23 Jun, 2020

Date

[illegible]

Harishankaran K

CTO, HackerRank

Department of Computer Engineering and Applications
GLA University, 17 km. Stone NH#2, Mathura-Delhi Road,
Chaumuha, Mathura – 281406 U.P (India)

ACKNOWLEDGEMENT

Presenting the ascribed project paper report in this very simple and official form, we would like to place my deep gratitude to GLA University for providing us the instructor Mr Abhishek Tiwari, our technical trainer and supervisor.

He has been helping us since Day 1 in this project. He provided us with the roadmap, the basic guidelines explaining on how to work on the project. He has been conducting regular meeting to check the progress of the project and providing us with the resources related to the project. Without his help, we wouldn't have been able to complete this project.

And at last but not the least we would like to thank our dear parents for helping us to grab this opportunity to get trained and also my colleagues who helped me find resources during the training.

Thanking You

Sign: *Dushyant Verma*	**Sign**: *Aryan Gupta*
Name of Candidate: Dushyant Verma	**Name of Candidate**: Aryan Gupta
University Roll No.:191500282	**University Roll No.**:191500157
Sign: *Jay Thakur*	**Sign**: *Abhishek Jadon*
Name of Candidate: Jay Thakur	**Name of Candidate**: Abhishek Jadon
University Roll No.:191500368	**University Roll No.**:191500026

ABSTRACT

In this project, we are creating an Web Application, basically a Courier Web App which we have named Drop Ship. This application will provide us a platform to access the shipment services and courier based services we want to give the ease of flexibility of shipment. All the users will be having their separate accounts on this app which will be connected to their email id. Any material that the user wish to deliver will be entered by him in the search box which works on the basis of queries input. Apart from searching the customer can add his item for shipment, the user can save the shipment detail he/she likes in the history bar. The app is suitable in the present scenario as the world is being digitalized then why not the courier system.. On the profile of the user, one can easily view the courier he/she has added. The web app will be completely efficient and transparent to the reviews of the people on the customer based ratings and its price. To get more details about the items one can click on the product that he/she wishes and get further grave details. This app will be using The Google MAP API for providing all the directions/routes. Further we use Stripe API all the necessary details of payment and transactions that the user may need about the convenience. The app also has a complete User Interface attached to the firebase a perfect login system with email id and password and a forget password too.

Web App ecosystem is diverse and is changing people’s life all over the world. Web users are expected to increase because of the advance changes of the operating system and the way it deals with issues and compatibility with other mobile devices. Furthermore, designing solutions for the problems that we may face in future is essential. Like this application definitely stands the need common people at any time at their fingertips without any barrier of place.

INTRODUCTION

1.1 CONTEXT

This Web Application "**Drop Ship**" has been submitted in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering at GLA University, Mathura supervised by Mr. Abhishek Tiwari. This project has been completed approximately two months and has been executed in modules, meetings have been organised to check the progress of the work and for instructions and guidelines.

1.2 MOTIVATION

In the recent years, we have realized the importance of courier services and how important it is for us to have our resources online. Couriers have been the greatest source of happiness to the people whom it get send, all the while and having them at the reach of our fingertips would be an opportunity hardly any people would afford to miss.

In the century we are living the world is progressing at a really great pace, a lot number of technologies come up every single day. To keep up with the technology is also important to survive in this world of digitalization and learning. Along with this we need to have a place to keep the resources for areas of our interest so we thought of developing a app which could provide us with virtual shipment as well as a platform where we could keep the customer record for marked.

Moreover, this kind of application can be used in areas where people can afford to send orders. This would be an excellent effort to provide flexibility of delivery without any boundaries to all.

1.3 OBJECTIVE

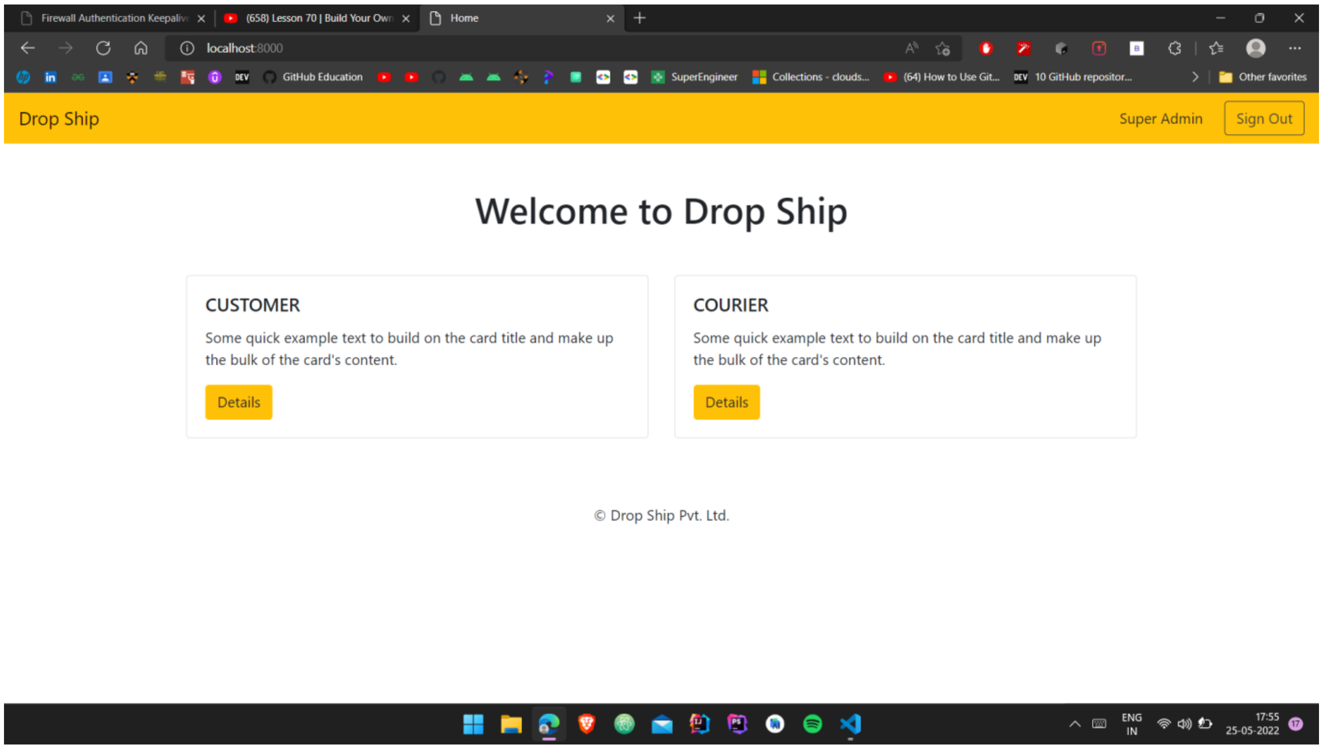
The main objective of this application is to create a Delivery app named "Drop Ship" which will have a lot of options and a space to keep up the parcel one wants to send. There will be a facility to search any courier one wishes to send by the use of any search bar like the shipment details and history. After the search there will be list of related options and one can view and read more about the details of the shipment and can further proceed it.

This application developed can be used at a variety of places, at vendor hubs and have its significance. The goal of the app was to provide a way to the consumers and users to get all the benefits they desire to send/receive at a particular location rather than randomly surfing the Internet.

1.4 EXISTING SYSTEM

In the present scenario, we are dealing with the virtual delivery system from thousands of other details present in Google Map API. With the help of this application, we are able to find a place where we can easily find the courier with the help of keywords. As this idea as already implemented here are some snaps how our application will look.

As soon as the user enters the web app, there will be landing home page containing the name of the app and then there will be a login /signup page. Initially there will be search bar as shown in the image below. Then on the basis of certain keywords the app will fetch the results and the items will be displayed as shown in the second screenshot.



Welcome to Drop Ship

CUSTOMER

Some quick example text to build on the card title and make up the bulk of the card's content.

COURIER

Some quick example text to build on the card title and make up the bulk of the card's content.

© Drop Ship Pvt. Ltd.

Figure-1: Existing System

1.5 SOURCES

The source of our project (including all the project work, documentations and presentations) will is available at the following

GitHub Repository Link

SOFTWARE REQUIREMENT ANALYSIS

Working Methodology:

- The purpose to make this project is to provide a platform where common people and vendor sign-up and assign their courier to us and our delivery company delivered that package at right time.
- For sigh-in and sign-up we have used Email password authentication.
- We have use google firebase phone authentication.
- Stripe payment API is used for payment purpose.
- Courier delivers packages to customers at home/work/offices
- Courier drives to designated area of distribution

HARDWARE AND SOFTWARE REQUIREMENTS

Software Requirements:

- **Operating System:** Windows
- **User Interface:** Html, CSS and Bootstrap
- **Internet:** 3G or above and WI-FI
- **Scripting:** JavaScript and Django
- **Language:** Python

Hardware Requirements:

- **RAM:** 4GB +
- **Storage:** 256GB +
- **Windows:** 7 Ultimate & Above

2.0 MODULES AND FUNCTIONALITIES

There are some functionalities we are going to implement that will enhance the performance of Drop Ship

1) **User Registration:** User must be able to register for the application through a valid E-mail. While using the application, user must be prompted to register their phone number with OTP verification.

2) **Order Booking:** User can come to the dashboard and choose the desired product with the desired specification and provide the pickup or delivery address which is then picked up by the courier and dispersed to customers.

3) **Payment:** After finalize the product user come to payment dashboard and fill their credentials for the secure payment generation.

4) **Track Order:** Package tracker is an all-in-one package tracking tool for all your shipments. We will remember all of your tracking numbers and pull delivery status information from dozens of carrier with extra features not offered on the carriers' sites.

5) **Delivery:** After doing all these exercises the product will be duly delivered before the time deadline.

2.1 Drop Ship Services:

People often state that they try to choose professional courier services that are the cheapest no matter what, but in general, people want to get the best quality and service for the amount that they pay, even if means spending more. Delivery is an important part of a purchase, so it's definitely something which any commercial establishment needs to pay attention to.

1) Reliability: The best courier service is always reliable in terms of getting your items safely delivered on time. There is nothing more frustrating for a business or a customer than having something important they expect to show up late.

2) Speed: Fast delivery matters the most. Customers trust courier services that timely delivers their parcels for smooth functioning of their businesses. Customers opt for courier companies that have daily, weekly or monthly routine deliveries that suit their needs.

3) Pricing: The best courier service understands good value for your business. Distribution of prices based on domestic and international deliveries is usually the best pricing option, since this helps to create transparency and an opportunity for customers to compare between their choices. Although, in the case of commercial couriers, sometimes asking for package or a custom deal is the quickest way to seal a value-based contract.

4) Communication: Maintaining contact with your customers in terms of delivery updates through various modes of communication like SMS, email and calls will help you build a long-lasting relationship with them.

5) Patience: Politeness in dealing with customers requires a smiling face and a positive attitude. These little things go a long way toward winning them over.

SOFTWARE DESIGN

Drop Ship Network:

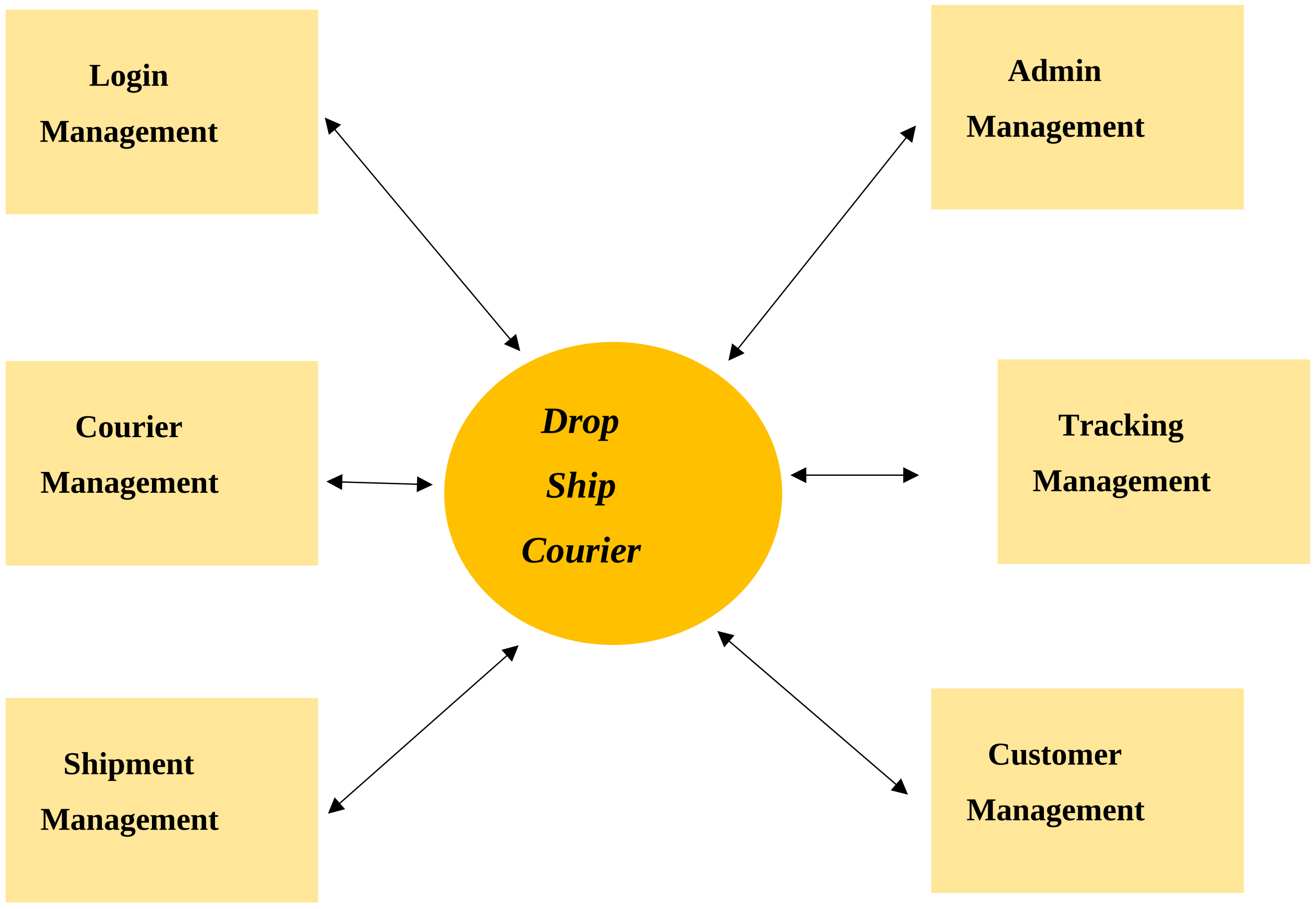


Figure: Working Figure

TECHNOLOGY USED

4. Django

- The origins of Django

Django was created in 2003 when web developers at the Lawrence Journal-World newspaper started using Python for their web development. After creating a number of websites, they started to factor out and reuse lots of common code and design patterns. That common code led to a generic web development framework that was open-sourced as the "Django" project in 2005. Since the original developers were surrounded by those newspaper writers, well-written documentation is a key part of Django. This means that there are excellent references to check out on the official Django documentation pages.

- **The Django community**

The Django framework is extremely large, but the Django community is absolutely massive. The community has contributed a lot of third party code for Django. No matter what we are trying to do, there's a good chance that we will find the solution for it on djangopackages.org. The website includes everything, from authentication and authorization to full-on Django-powered content management systems, from e-commerce add-ons to integrations with Stripe.

- **Django features**

Some features that make Django an ideal framework for web application development are as follows:

- Super fast: Django development is extremely fast. Our ideas can take the shape of a product very quickly.
- Fully loaded: Django has dozens of projects that can be integrated to carry out common tasks such as user authentication, authorization, and content administration.
- Versatile: Django can be used for almost any kind of project, from CMSs to e-commerce apps to on-demand delivery platforms.
- Secure: Django also has support to prevent common security issues, including cross-site request forgery, cross-site scripting, SQL injection, and clickjacking.
- Scalable: Django websites can scale fast to meet high traffic demands.

4.1 Python

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

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

4.2 Bootstrap

- Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.
- It is absolutely free to download and use.
- It is a front-end framework used for easier and faster web development.
- It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others.
- It can also use JavaScript plug-ins.
- It facilitates you to create responsive designs.

• History of Bootstrap

Bootstrap was developed by Mark Otto and Jacob Thornton at Twitter. It was released as an open source product in August 2011 on GitHub.
In June 2014 Bootstrap was the No.1 project on GitHub.

• Why use Bootstrap

Following are the main advantage of Bootstrap:

- It is very easy to use. Anybody having basic knowledge of HTML and CSS can use Bootstrap.
- It facilitates users to develop a responsive website.
- It is compatible on most of browsers like Chrome, Firefox, Internet Explorer, Safari and Opera etc.

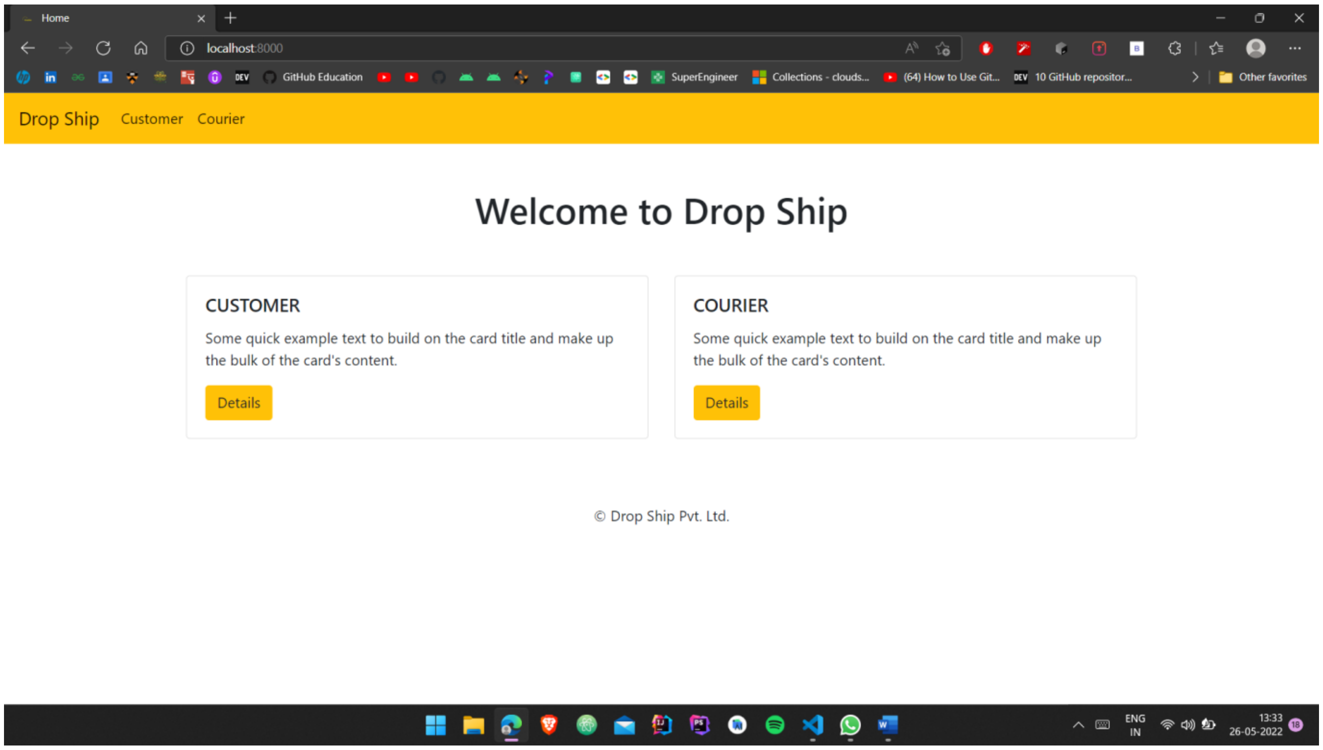
Implementation Of Activities:

The implementation of the activities can be separated into five major parts:

1. Customer Sign-up/sign-in
2. Admin Dashboard
3. Payment Activity
4. Track courier
5. Create courier

5.2 User Interface

- **Splash Screen**



Drop Ship Customer Courier

Welcome to Drop Ship

CUSTOMER

Some quick example text to build on the card title and make up the bulk of the card's content.

Details

COURIER

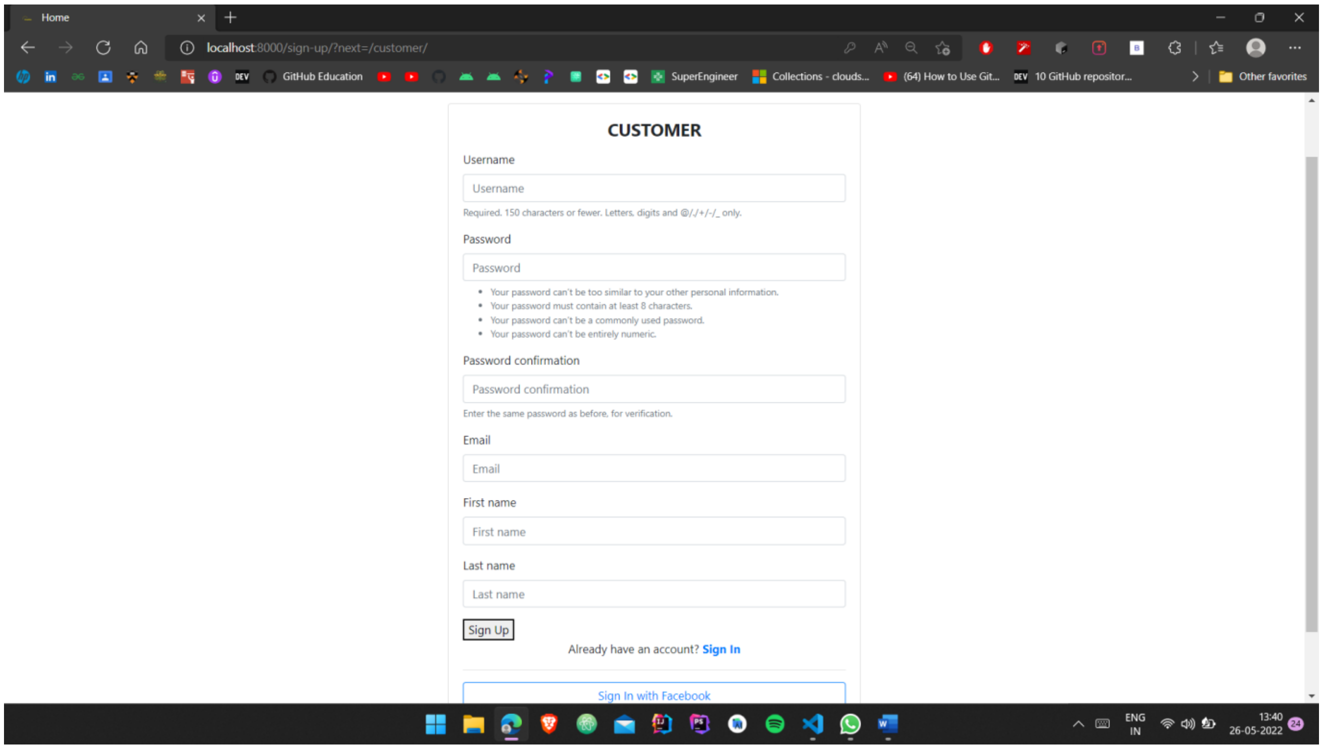
Some quick example text to build on the card title and make up the bulk of the card's content.

Details

© Drop Ship Pvt. Ltd.

Figure-1: Splash Screen

- **Register Page**



CUSTOMER

Username

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

Password

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation

Enter the same password as before, for verification.

Email

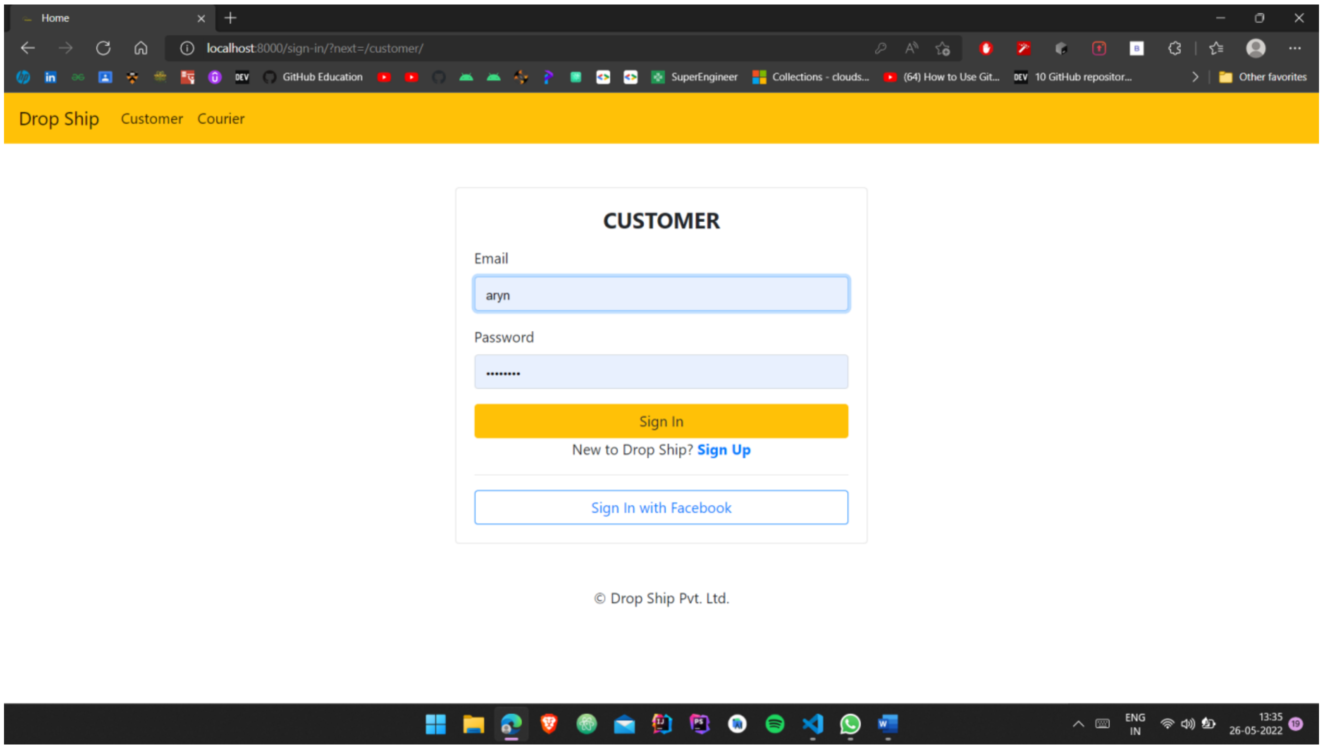
First name

Last name

Already have an account? Sign In

Figure-2: Register Page

- **Login Page**



CUSTOMER

Email: aryn

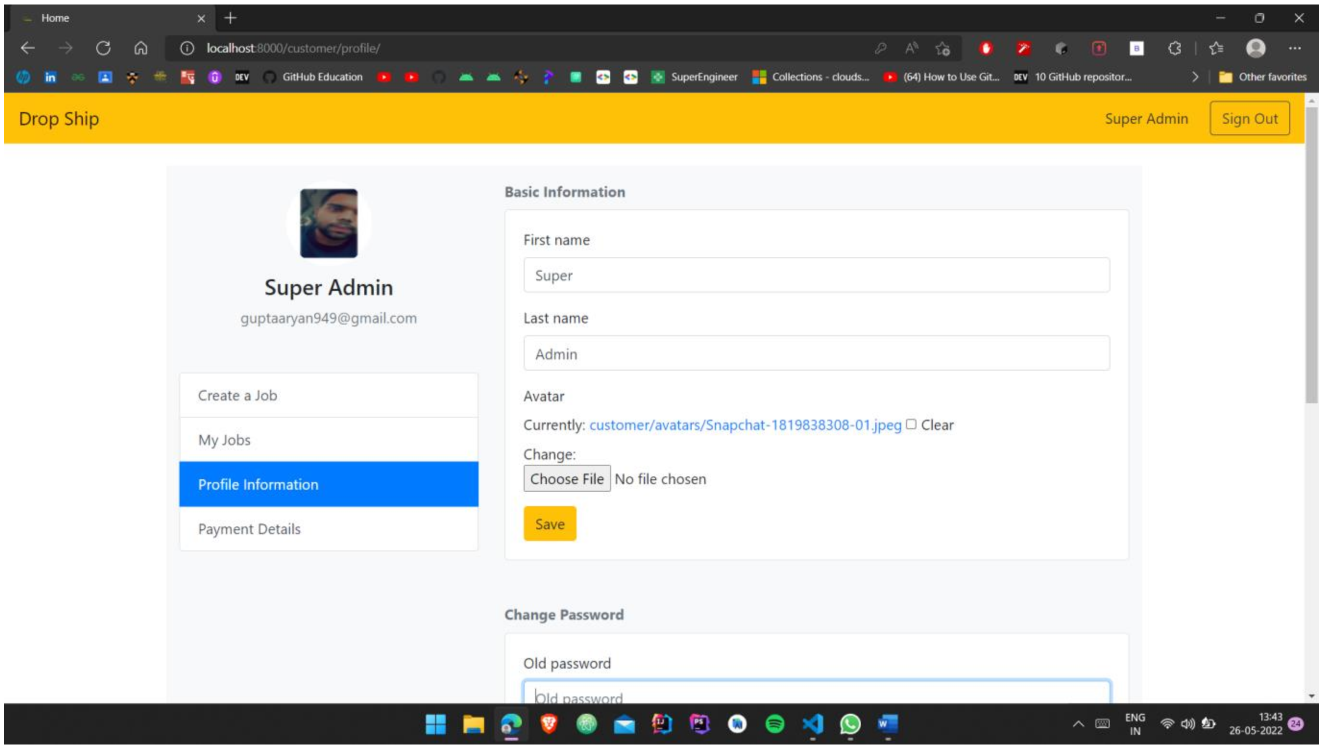
Password: ••••••••

New to Drop Ship? Sign Up

© Drop Ship Pvt. Ltd.

Figure-3: Login Page

- **Profile Page:**



First name

Super

Last name

Admin

Avatar

Currently: customer/avatars/Snapchat-1819838308-01.jpeg ☐ Clear

Change:

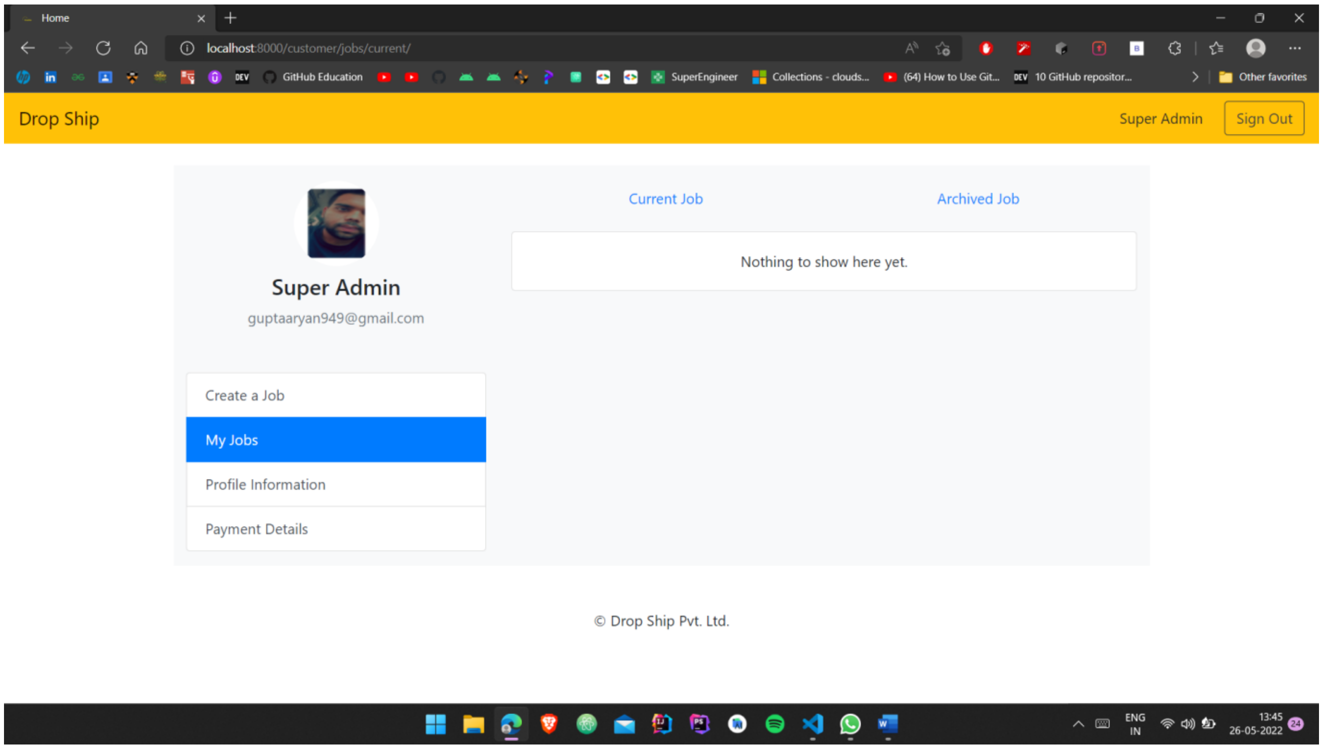
Choose File No file chosen

Old password

Old password

Figure-4: Profile Page

- **Create a Job:**



Nothing to show here yet.

Figure-5: Create a Job

- **Payment Page:**

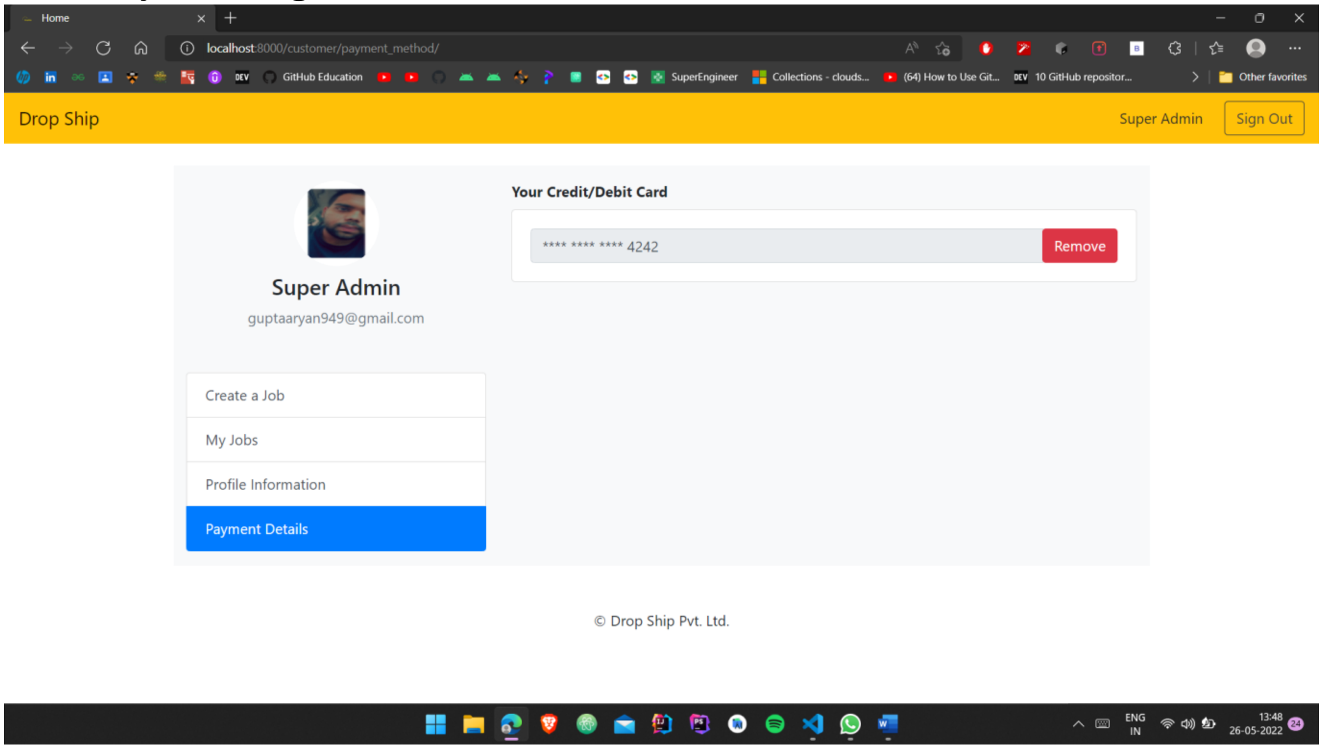
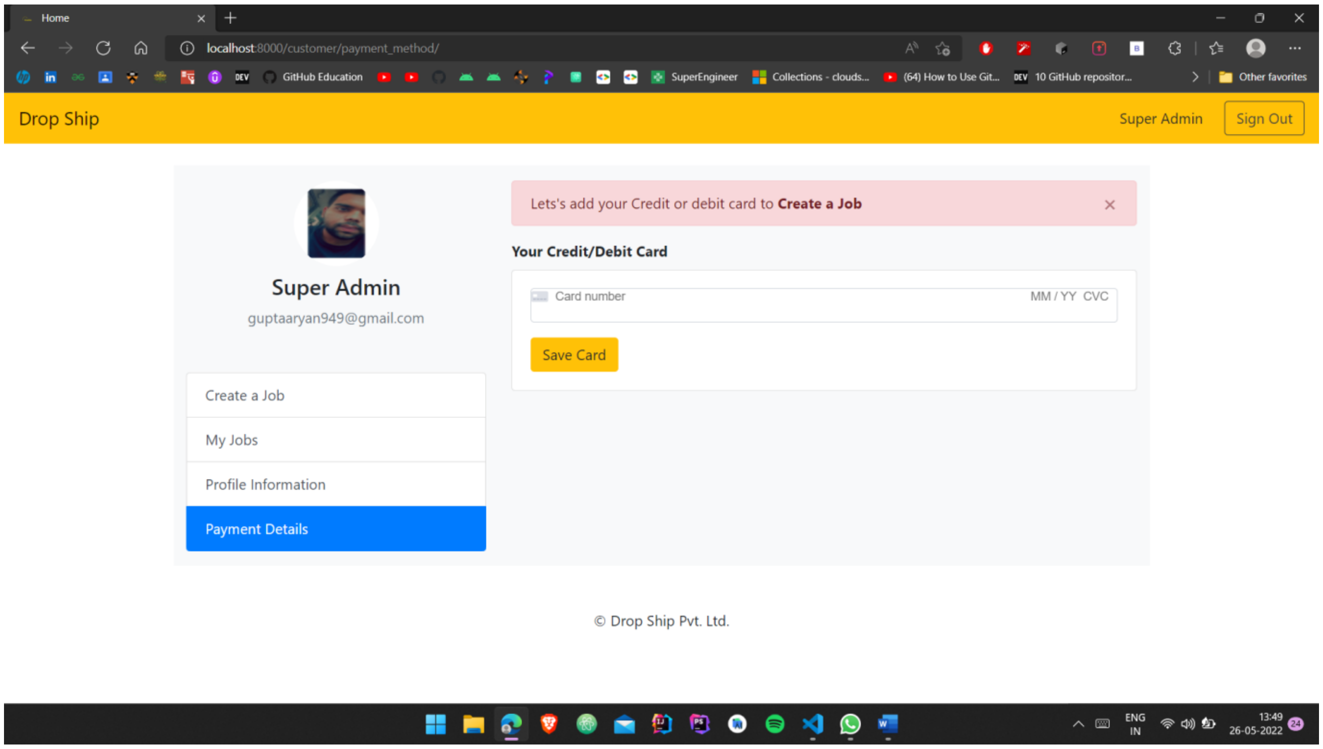


Figure-6: Payment Page

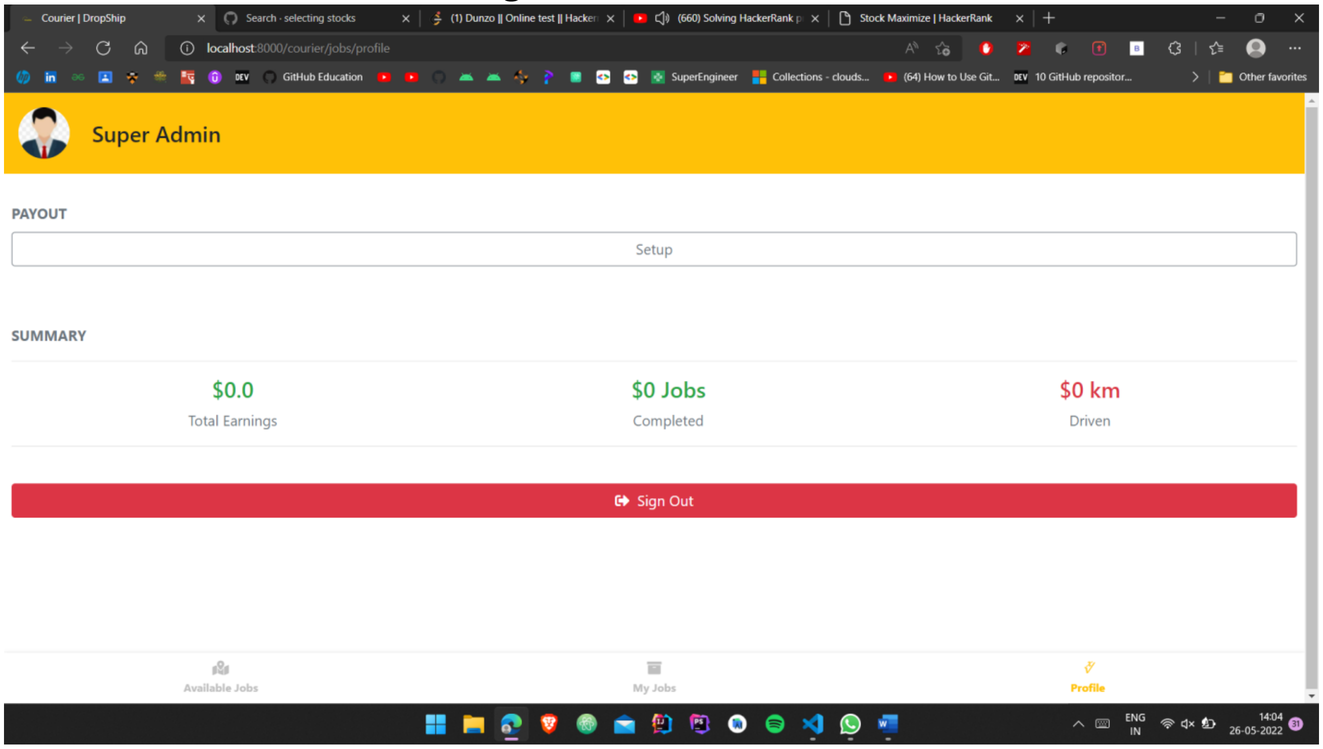
- **Add Debit/Credit Card:**



Lets's add your Credit or debit card to **Create a Job**

Figure-7: Add Debit/Credit Card

- **Courier Confirmation Page:**



Super Admin

PAYOUT

Setup

SUMMARY

$0.0 Total Earnings

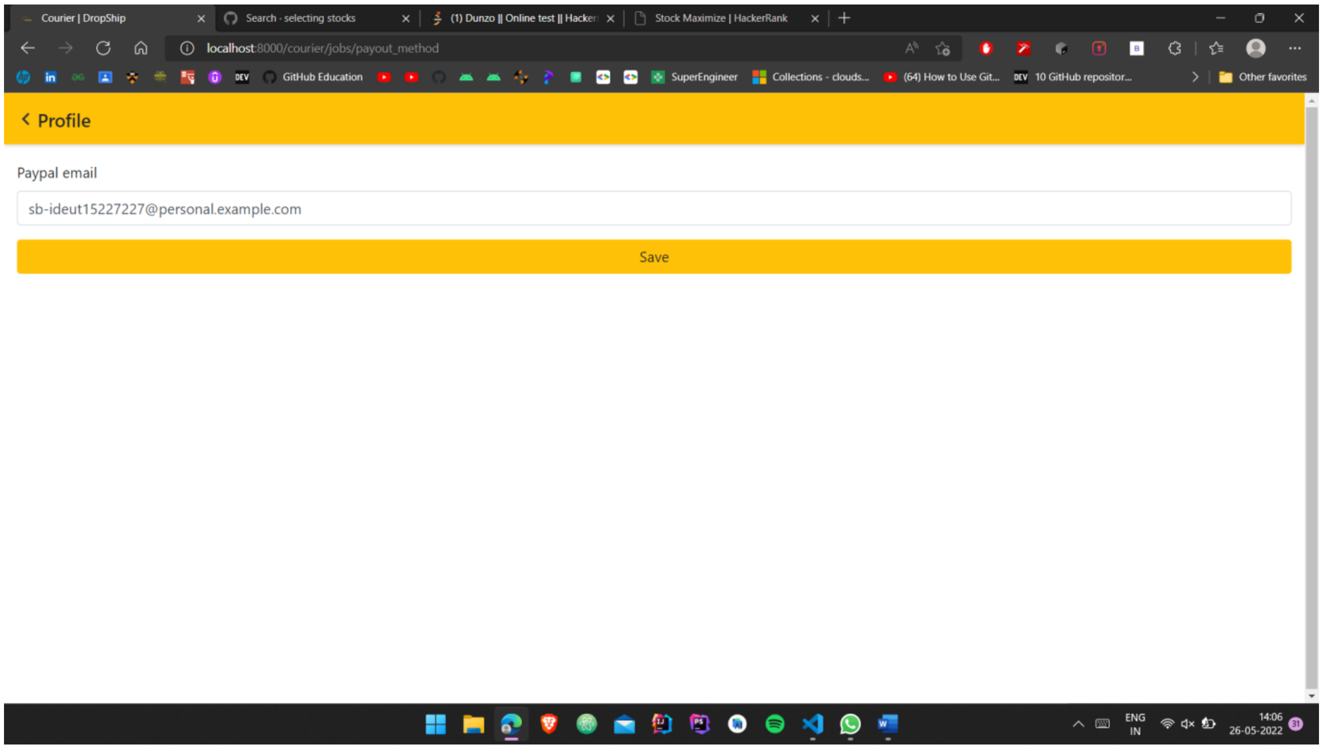
$0 Jobs Completed

$0 km Driven

Sign Out

Figure-8: Courier Confirmation Page

- **PayPal-email:**



Profile

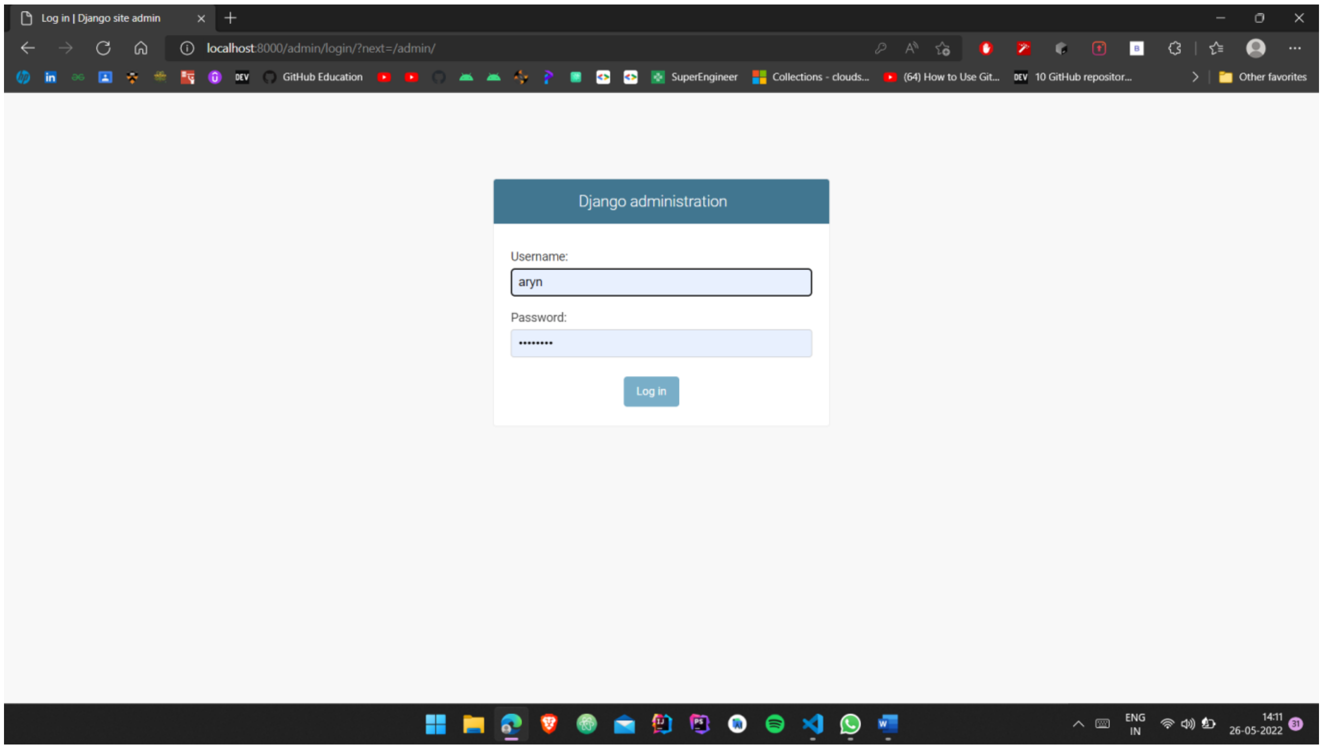
Paypal email

sb-ideut15227227@personal.example.com

Save

Figure-9: PayPal-email

- **Admin Side**



Django administration

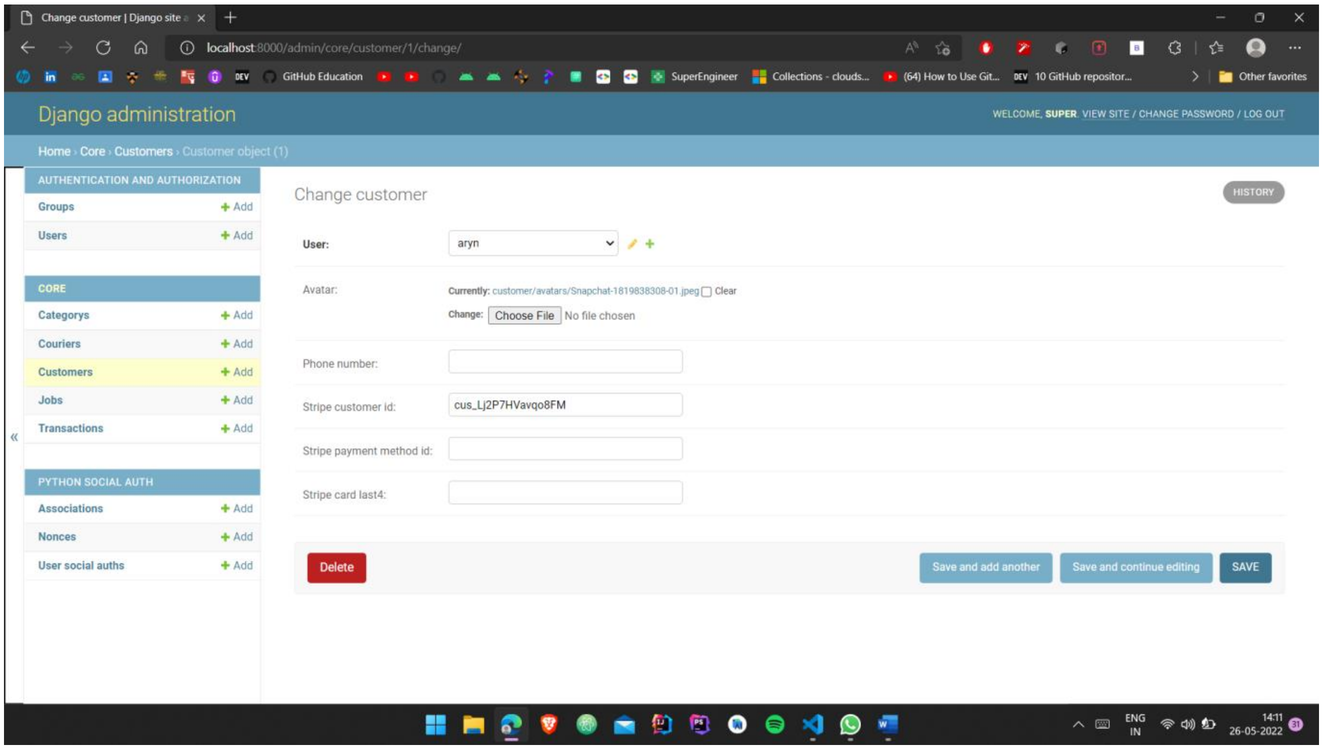
Username: aryn

Password: ••••••••

Log in

Figure-10: Admin Side

- **Customer Details:**



Change customer

User: aryn

Avatar: Currently: customer/avatars/Snapchat-1819838308-01.jpeg ☐ Clear
Change: Choose File No file chosen

Phone number:

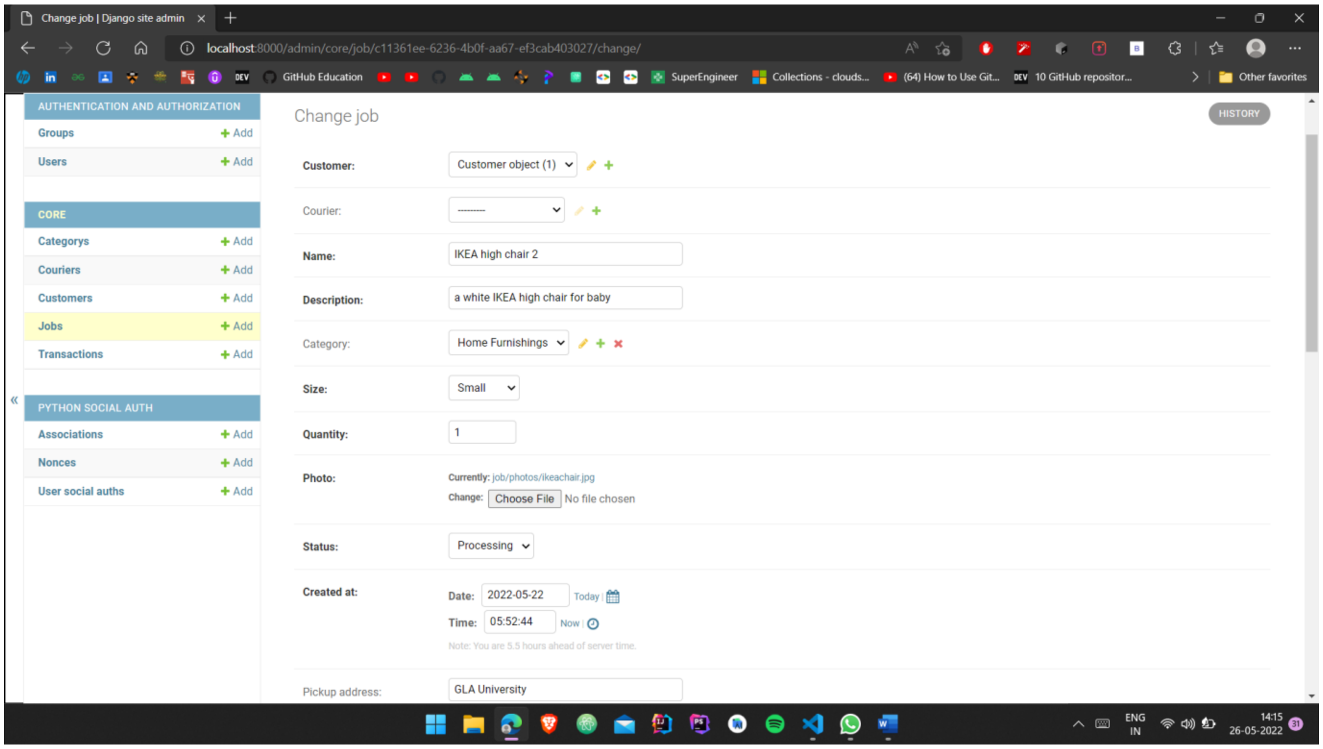
Stripe customer id: cus_Lj2P7HVavqo8FM

Stripe payment method id:

Stripe card last4:

Figure-11: Customer Details

- **Job Details:**

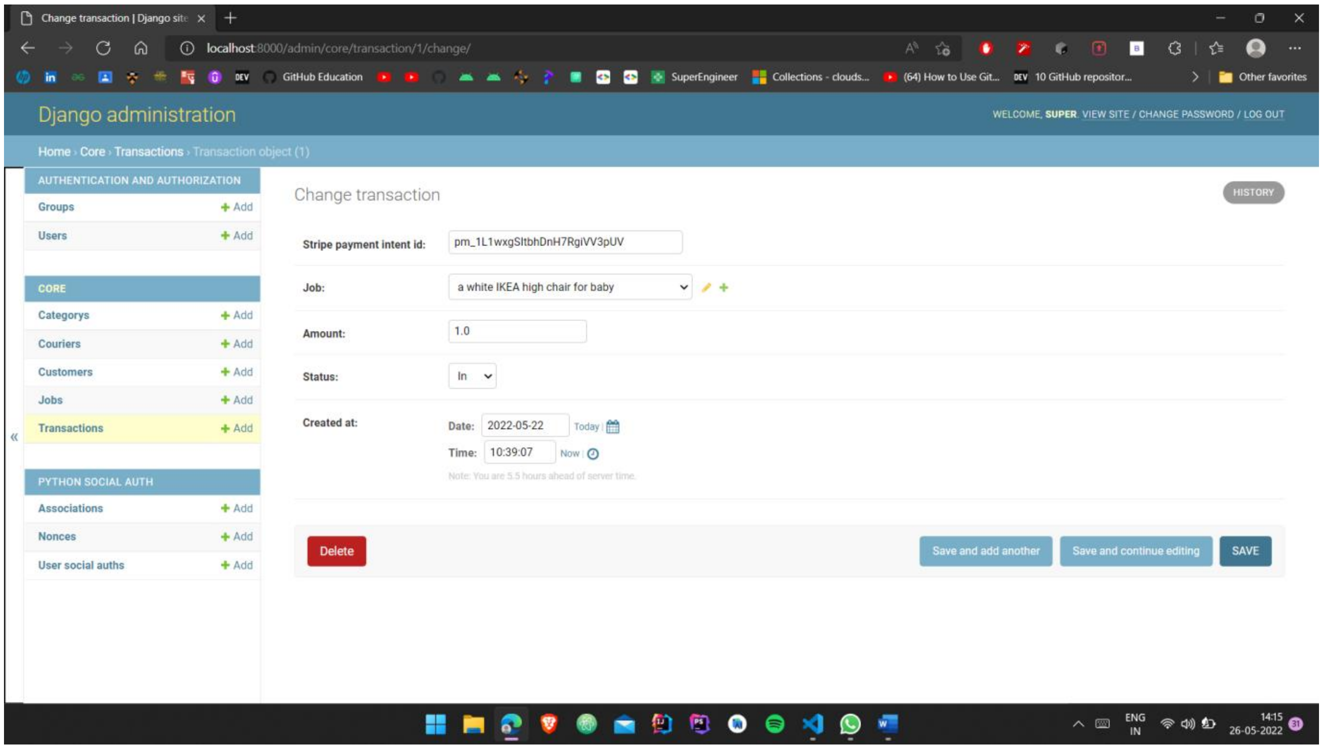


Change job

Customer:	Customer object (1)
Courier:	---------
Name:	IKEA high chair 2
Description:	a white IKEA high chair for baby
Category:	Home Furnishings
Size:	Small
Quantity:	1
Photo:	Currently: job/photos/ikeachair.jpg Change: Choose File No file chosen
Status:	Processing
Created at:	Date: 2022-05-22 Time: 05:52:44
Pickup address:	GLA University

Figure-12: Job Details

- **Transactions:**

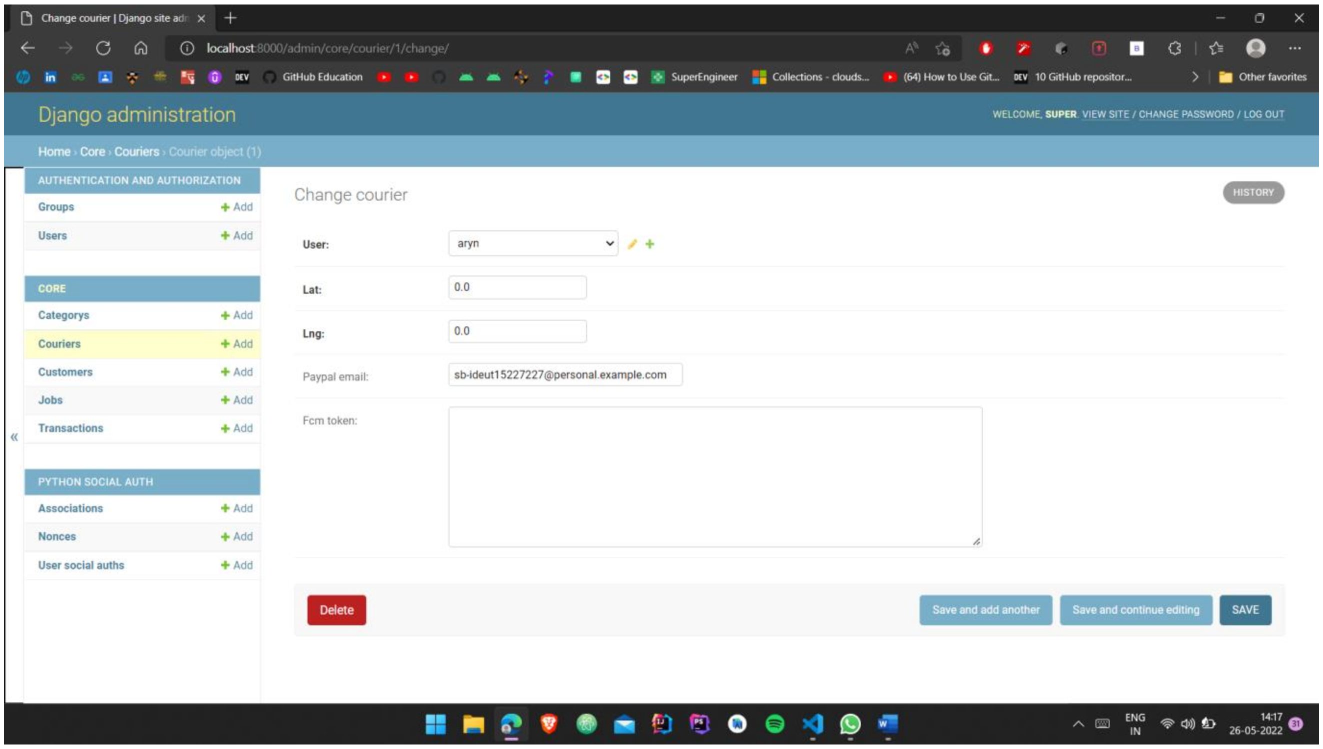


Change transaction

Stripe payment intent id:	pm_1L1wxgSltbhDnH7RgiVV3pUV
Job:	a white IKEA high chair for baby
Amount:	1.0
Status:	In
Created at:	Date: 2022-05-22 Time: 10:39:07

Figure-13: Transactions

- **Couriers:**


Change courier | Django site adm

localhost:8000/admin/core/courier/1/change/

Django administration

WELCOME, SUPER. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Core › Couriers › Courier object (1)

AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add

CORE
Categorys + Add
Couriers + Add
Customers + Add
Jobs + Add
Transactions + Add

PYTHON SOCIAL AUTH
Associations + Add
Nonces + Add
User social auths + Add

Change courier

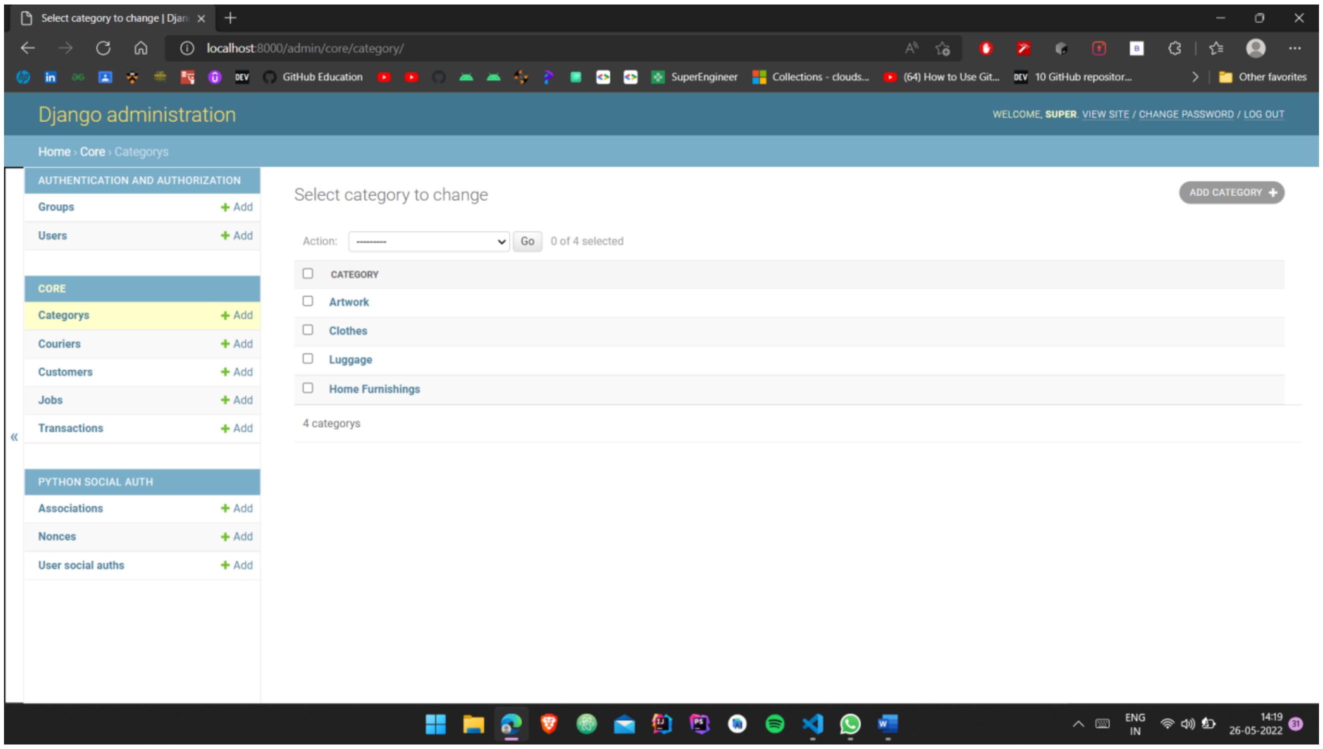
HISTORY

User:	aryn
Lat:	0.0
Lng:	0.0
Paypal email:	sb-ideut15227227@personal.example.com
Fcm token:	

Delete | Save and add another | Save and continue editing | SAVE

Figure-14: Couriers

- **Category's:**


Select category to change | Djan

localhost:8000/admin/core/category/

Django administration

WELCOME, SUPER. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Core › Categorys

AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add

CORE
Categorys + Add
Couriers + Add
Customers + Add
Jobs + Add
Transactions + Add

PYTHON SOCIAL AUTH
Associations + Add
Nonces + Add
User social auths + Add

Select category to change

ADD CATEGORY +

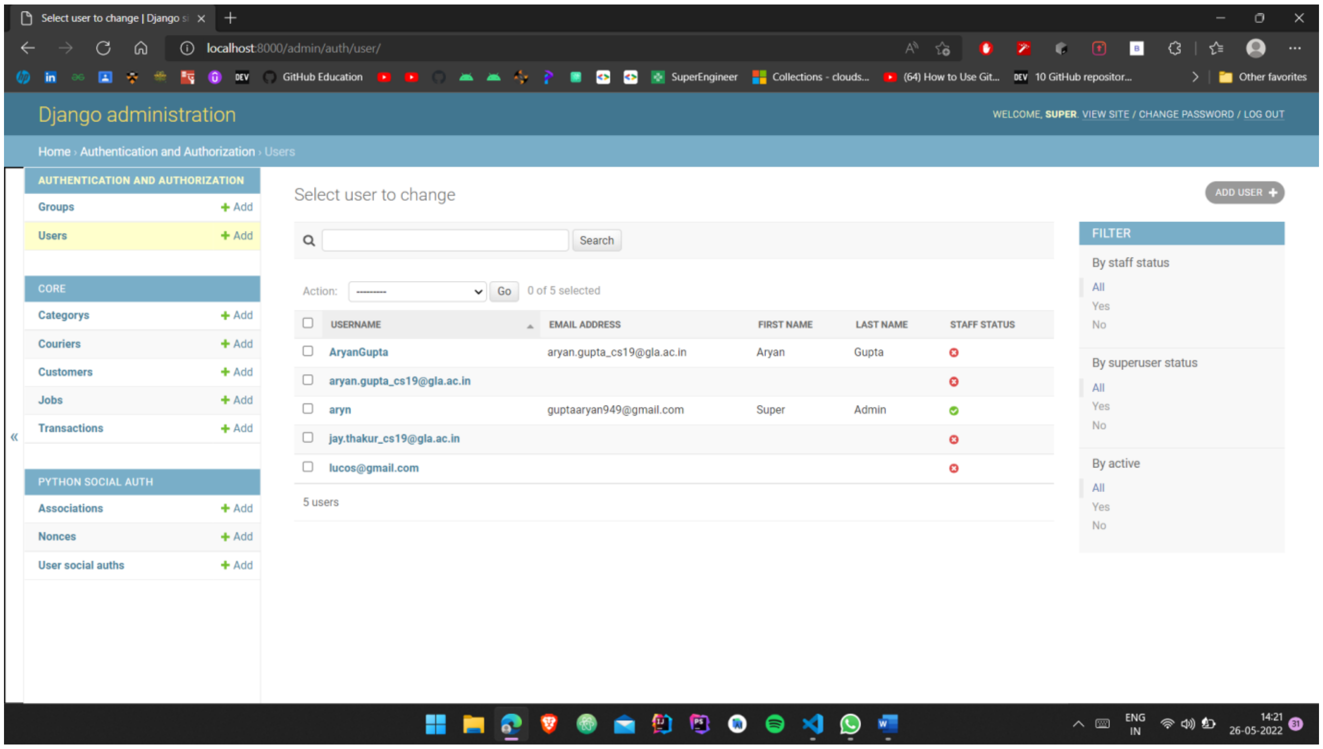
Action: --------- Go 0 of 4 selected

CATEGORY
Artwork
Clothes
Luggage
Home Furnishings

4 categorys

Figure-15: Category's

- **Users:**


Django administration

WELCOME, SUPER. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Authentication and Authorization › Users

AUTHENTICATION AND AUTHORIZATION: Groups + Add; Users + Add

CORE: Categorys + Add; Couriers + Add; Customers + Add; Jobs + Add; Transactions + Add

PYTHON SOCIAL AUTH: Associations + Add; Nonces + Add; User social auths + Add

Select user to change

Action: --------- Go 0 of 5 selected

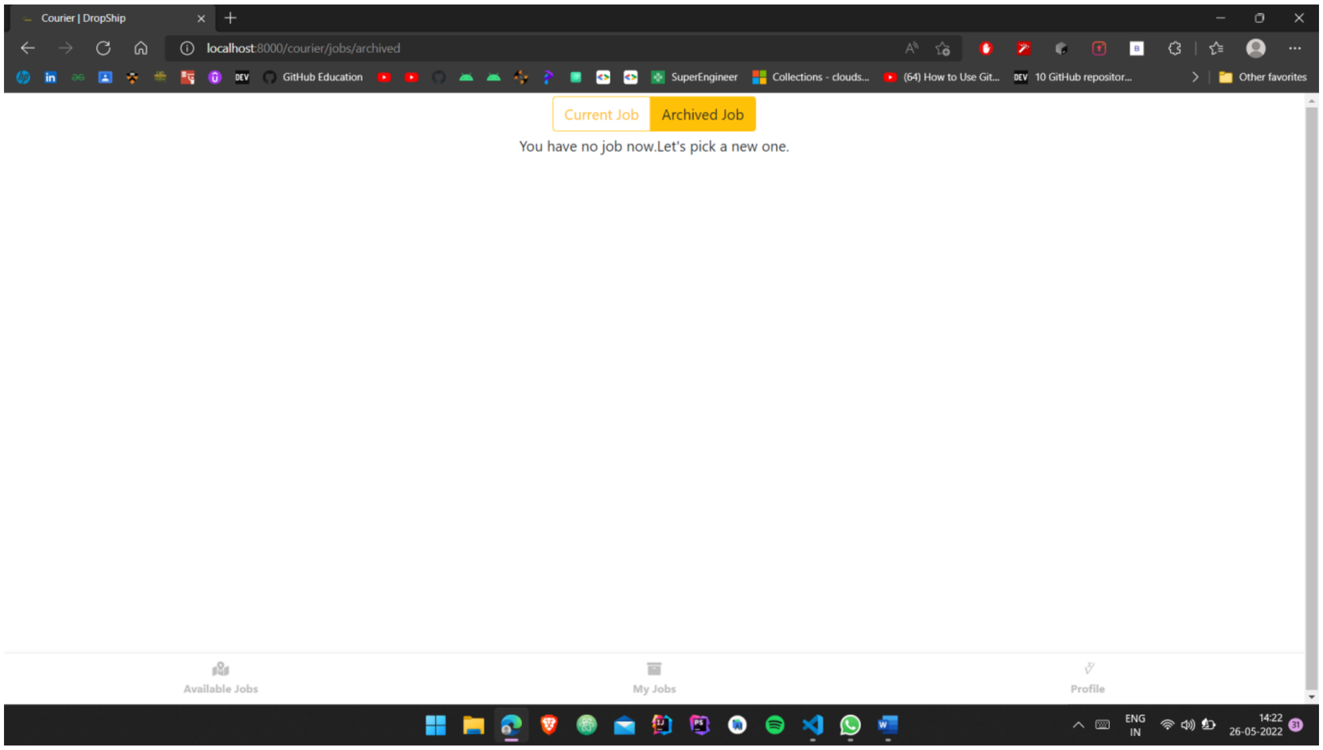
USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
AryanGupta	aryan.gupta_cs19@gla.ac.in	Aryan	Gupta	✗
aryan.gupta_cs19@gla.ac.in				✗
aryn	guptaaryan949@gmail.com	Super	Admin	✓
jay.thakur_cs19@gla.ac.in				✗
lucos@gmail.com				✗

5 users

FILTER — By staff status: All, Yes, No. By superuser status: All, Yes, No. By active: All, Yes, No.

Figure-16: Users

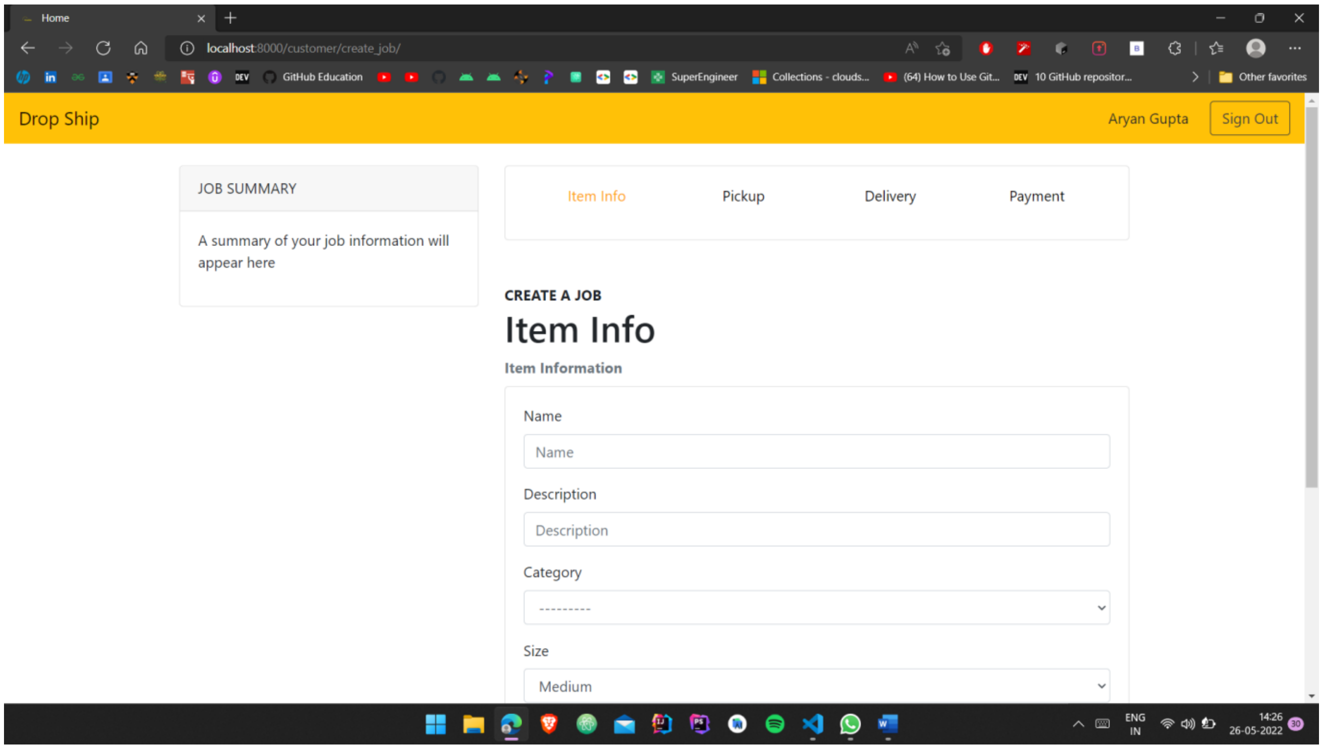
- **All Jobs:**


localhost:8000/courier/jobs/archived

You have no job now.Let's pick a new one.

Figure-17: All Jobs

- **Item Info:**


Drop Ship
Aryan Gupta Sign Out

JOB SUMMARY

A summary of your job information will appear here

Item Info | Pickup | Delivery | Payment

CREATE A JOB

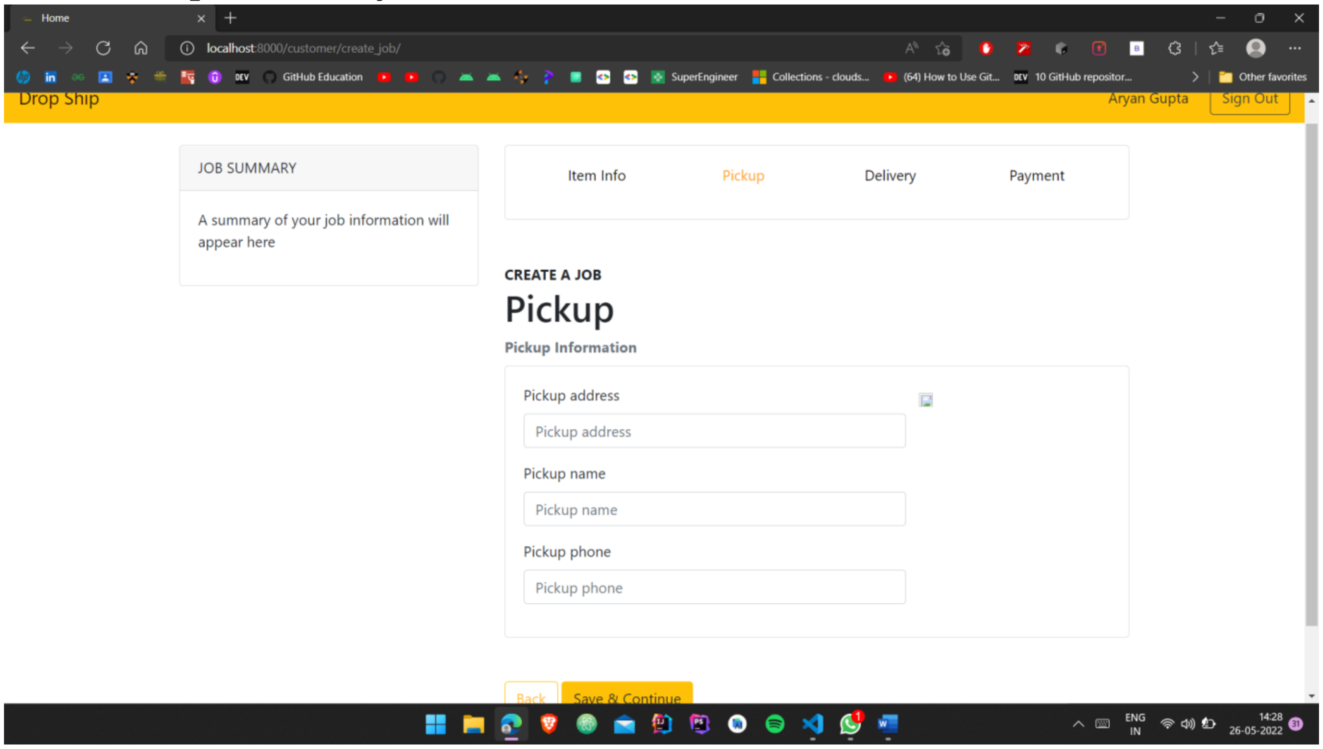
Item Info

Item Information

Name: Name
Description: Description
Category: ---------
Size: Medium

Figure-18: Item Info

- **Pickup & Delivery Address**:


Drop Ship
Aryan Gupta Sign Out

JOB SUMMARY

A summary of your job information will appear here

Item Info | Pickup | Delivery | Payment

CREATE A JOB

Pickup

Pickup Information

Pickup address: Pickup address
Pickup name: Pickup name
Pickup phone: Pickup phone

Back | Save & Continue

Figure-19: Pickup & Delivery Address

CONCLUSION

Proposed Drop Ship App is an android application that will allow users to delivery of goods by subject name. This application takes in a user input and searches the Google Maps API with the user input and gets a list of routes based on the users search query. Search result screen will contain a list of recent items with details: average, rating Price of the parcel. To get the information of the particular good of user that can be click upon the item from the list and then will be taken to the new tab where description and other information related to the product will be available.

REFERENCES

1. **Bootstrap**
2. **Django**
3. **Stripe**
4. **PayPal**
5. **Stack Overflow**
6. **Firebase**
7. **Google Map API**

Code Activity:

Admin.py:

```python
import random
import string
from django.contrib import admin, messages
from django.conf import settings
from paypalrestsdk import configure, Payout
from . models import *

configure({
    "mode":settings.PAYPAL_MODE,
    "client_id":settings.PAYPAL_CLIENT_ID,
    "client_secret":settings.PAYPAL_CLIENT_SECRET,
})

def payout_to_courier(modeladmin,request,queryset):
    payout_items = []
    transaction_querysets = []

    #Step1 - Get all the valid couriers in queryset
    for courier in queryset:
        if courier.paypal_email:
            courier_in_transactions = Transaction.objects.filter(
                job__courier = courier,
                status = Transaction.IN_STATUS
            )
            if courier_in_transactions:
                transaction_querysets.append(courier_in_transactions)
                balance = sum(i.amount for i in courier_in_transactions)
                payout_items.append({
                    "recipient_type": "EMAIL",
                    "amount": {
                        "value": "{:.2f}".format(balance * 0.8),
                        "currency": "USD"
                    },
                    "receiver": courier.paypal_email,
                    "note": "Thank you.",
                    "sender_item_id": str(courier.id)
                })
    #Step2 - Send payout batch + email to receivers
    sender_batch_id = ''.join(random.choice(string.ascii_uppercase) for i in 
range(12))
    payout = Payout({
      "sender_batch_header": {
        "sender_batch_id": sender_batch_id,
        "email_subject": "You have a payment"
        },
        "items": payout_items
    })
    #Step3 - Execute payout process and Update the transaction status to "OUT" if 
success
    try:
        if payout.create():
            for t in transaction_querysets:
                t.update(status = Transaction.OUT_STATUS)
            messages.success(request, "payout[%s] created successfully" % 
(payout.batch_header.payout_batch_id))
        else:
            messages.error(request, payout.error)
    except Exception as e:
        messages.error(request, str(e))

payout_to_courier.short_description="Payout to Couriers"

class CourierAdmin(admin.ModelAdmin):
    list_display=['user','paypal_email','balance']
    actions = [payout_to_courier]

    def user_full_name(self, obj):
        return obj.user.get_full_name()

    def balance(self,obj):
        return round( sum (t.amount for t in  Transaction.objects.filter(job__courier 
= obj, status=Transaction.IN_STATUS)) *0.8, 2)


class TransactionAdmin(admin.ModelAdmin):
    list_display=['stripe_payment_intent_id','courier_paypal_email', 'customer', 
'courier', 'job','amount','created_at']

    def customer(self,obj):
        return obj.job.customer
    def courier(self,obj):
        return obj.job.courier
    def courier_paypal_email(sefl,obj):
        return obj.job.courier.paypal_email if obj.job.courier else None
#Register your models here.
admin.site.register(Customer)
admin.site.register(Courier, CourierAdmin)
admin.site.register(Category)
admin.site.register(Job)
admin.site.register(Transaction, TransactionAdmin)
```

Settings.py:

```
"""
Django settings for dropship project.

Generated by 'django-admin startproject' using Django 3.1.3.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.1/ref/settings/
"""

from pathlib import Path
from pickle import TRUE
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '^46vzw_8*s^fqce94^afy_13f*)gqsf@_=o=eft_7_iz_&ccoi'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['*',]


# Application definition
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'bootstrap4',
    'social_django',
    'core.apps.CoreConfig',
    'channels',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'core.middleware.ProfileMiddlware',

]

ROOT_URLCONF = 'dropship.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'social_django.context_processors.backends',
                'social_django.context_processors.login_redirect',
            ],
        },
    },
]
```

```
WSGI_APPLICATION = 'dropship.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.1/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'

LOGIN_URL = '/sign-in'
LOGIN_REDIRECT_URL = '/'

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = "/media/"

AUTHENTICATION_BACKENDS = (
    'social_core.backends.facebook.FacebookOAuth2',
    'django.contrib.auth.backends.ModelBackend',
)


SOCIAL_AUTH_FACEBOOK_KEY = "392295216244044"
SOCIAL_AUTH_FACEBOOK_SECRET = "36744e1065e0bfafe5d94d09d286e35b"
SOCIAL_AUTH_FACEBOOK_SCOPE = ['email']
SOCIAL_AUTH_FACEBOOK_PROFILE_EXTRA_PARAMS = {
    'fields': 'id, name, email'
}

EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_USE_TLS = TRUE
EMAIL_PORT = 587
EMAIL_HOST_USER = 'aryan.gupta_cs19@gla.ac.in'
EMAIL_HOST_PASSWORD = 'Aryan@1234'
DEFAULT_FROM_EMAIL = 'Drop Ship <no-reply@dropship.localhost>'

FIREBASE_ADMIN_CREDENTIAL = os.path.join(BASE_DIR, "dropship-f1ad9-firebase-adminsdk-
sx5ku-071bcbe094.json")
STRIPE_API_PUBLIC_KEY =
"pk_test_51L1W4mSItbhDnH7RuA5qbViqyR1iGOoBrdNXoSdgFRbz8VJT5YFssVW3Qj24ep4eLuE3pEvnyfH
xDBXBgLV9FKfZ00of9MoXEs"
STRIPE_API_SECRET_KEY =
"sk_test_51L1W4mSItbhDnH7REM6oGGUx5GwmSsifwfJJkIe77ZlphODl90Yzyel2cLFy85C0haGrC1IWbbN
pjjQhpSewLand008kGZzq36"
GOOGLE_MAP_API_KEY = "AIzaSyDnFFBQo9GkvDdSgjf2T-r7z8LdaXYG_LA"
PAYPAL_MODE = "sandbox"
PAYPAL_CLIENT_ID = "AUN2QyvcrQDuVqx8ZK_g9sNi9s-
d2JhvvvgMee_HfofS_HVwGLvyzI6Kr0RAeEpameakW3UXNc5rQ8vI"
PAYPAL_CLIENT_SECRET = "ENGaP3bR5yApwHddJpYfG-PPd2r-vjToCHpB4DIZ-
YD7i_6dvUIZgGnF2RYWFoSoSIdMeElHsystX6d-"
```

```
NOTIFICATION_URL = ""
ASGI_APPLICATION = "dropship.asgi.application"

# Channels

CHANNEL_LAYERS = {
    'default': {
        'BACKEND': 'channels_redis.core.RedisChannelLayer',
        'CONFIG': {
            "hosts": [('127.0.0.1', 6379)],
        },
    },
}
```

Urls.py:

```
from django.views.generic import TemplateView
from venv import create
from django.contrib import admin
from django.urls import path, include
from django.contrib.auth import views as auth_views
from django.conf import settings
from django.conf.urls.static import static
from core import views, consumers

from core.customer import views as customer_views
from core.courier import views as courier_views, apis as courier_apis

customer_urlspatterns =[
    path('', customer_views.home, name="home"),
    path('profile/', customer_views.profile_page, name="profile"),
    path('payment_method/', customer_views.payment_method_page,
name="payment_method"),
    path('create_job/', customer_views.create_job_page, name="create_job"),
    path('jobs/current/', customer_views.current_jobs_page, name="current_jobs"),
    path('jobs/archived/', customer_views.archived_jobs_page, name="archived_jobs"),
    path('jobs/<job_id>/', customer_views.job_page, name="job"),


]


courier_urlspatterns =[
    path('', courier_views.home, name="home"),
    path('jobs/available/', courier_views.available_jobs_page,
name="available_jobs"),
    path('jobs/available/<id>/', courier_views.available_job_page,
name="available_job"),
    path('jobs/current/', courier_views.current_job_page, name="current_job"),
    path('jobs/current/<id>/take_photo', courier_views.current_job_take_photo_page,
name="current_job_take_photo"),
    path('jobs/complete', courier_views.job_complete_page, name="job_complete"),
    path('jobs/archived', courier_views.archived_jobs_page, name="archived_jobs"),
    path('jobs/profile', courier_views.profile_page, name="profile"),
    path('jobs/payout_method', courier_views.payout_method_page,
name="payout_method"),
    path('api/jobs/available/', courier_apis.available_jobs_api, name =
"available_jobs_api"),
    path('api/jobs/current/<id>/update/', courier_apis.current_job_update_api, name =
"current_job_update_api"),
    path('api/fcm-token/update/', courier_apis.fcm_token_update_api, name =
"fcm_token_update_api"),

]


urlpatterns = [
    path('admin/', admin.site.urls),

    path('', include('social_django.urls', namespace='social')),
    path('', views.home),

    path('sign-in/', auth_views.LoginView.as_view(template_name="sign_in.html")),
    path('sign-out/', auth_views.LogoutView.as_view(next_page="/")),
    path('sign-up/', views.sign_up),

    path('customer/', include((customer_urlspatterns, 'customer'))),
    path('courier/', include((courier_urlspatterns, 'courier'))),
    path('firbase-messaging-sw.js',(TemplateView.as_view(template_name="firebase-
messaging-sw.js", content_type = "application/javascript",))),

]

websocket_urlpatterns = [
    path('ws/jobs/<job_id>/', consumers.JobConsumer.as_asgi())
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root = settings.MEDIA_ROOT)
```

Customer Class:

Forms.py:

```
from dataclasses import fields
from pyexpat import model
from django import forms
from django.contrib.auth.models import User

from core.models import Customer, Job
class BasicUserForm(forms.ModelForm):
    class Meta:
        model = User
        fields = ('first_name', 'last_name')

class BasicCustomerForm(forms.ModelForm):
    class Meta:
        model = Customer
        fields = ('avatar',)

class JobCreatedStep1Form(forms.ModelForm):
    class Meta:
        model = Job
        fields = ('name', 'description', 'category', 'size', 'quantity', 'photo')


class JobCreatedStep2Form(forms.ModelForm):
    pickup_address = forms.CharField(required=True)
    pickup_name = forms.CharField(required=True)
    pickup_phone = forms.CharField(required=True)
    class Meta:
        model = Job
        fields = ('pickup_address', 'pickup_lat', 'pickup_lng', 'pickup_name', 
'pickup_phone')

class JobCreatedStep3Form(forms.ModelForm):
    delivery_address = forms.CharField(required=True)
    delivery_name = forms.CharField(required=True)
    delivery_phone = forms.CharField(required=True)
    class Meta:
        model = Job
        fields = ('delivery_address', 'delivery_lat', 'delivery_lng',
'delivery_name', 'delivery_phone')
```

Views.py:

```
from ast import Expression
from turtle import distance
from urllib import response
import stripe
import requests
import firebase_admin
from firebase_admin import credentials, auth, messaging
from audioop import reverse
from django.shortcuts import redirect, render
from django.contrib.auth.decorators import login_required
from core.customer import forms
from django.contrib.auth.forms import PasswordChangeForm
from django.urls import reverse
from django.contrib.auth import update_session_auth_hash
from django.contrib import messages
from django.conf import settings

from core.models import Customer, Job, Transaction
from dropship.settings import NOTIFICATION_URL

cred = credentials.Certificate(settings.FIREBASE_ADMIN_CREDENTIAL)
firebase_admin.initialize_app(cred)

stripe.api_key = settings.STRIPE_API_SECRET_KEY

@login_required()
def home(request):
    return redirect(reverse('customer:profile'))


@login_required(login_url='/sign-in/?next=/customer/')
def profile_page(request):
    user_form = forms.BasicUserForm(instance=request.user)
    customer_form = forms.BasicCustomerForm(instance=request.user.customer)
    password_form = PasswordChangeForm(request.user)

    if request.method == "POST":
        if request.POST.get('action') == 'update_profile':
            user_form = forms.BasicUserForm(request.POST, instance=request.user)
            customer_form = forms.BasicCustomerForm(request.POST, request.FILES,
instance=request.user.customer)

            if user_form.is_valid() and customer_form.is_valid():
                user_form.save()
                customer_form.save()

                messages.success(request, "Your profile has been updated")
                return redirect(reverse('customer:profile'))

        elif request.POST.get('action') == 'update_password':
            password_form=PasswordChangeForm(request.user, request.POST)
            if password_form.is_valid():
                user=password_form.save()
                update_session_auth_hash(request, user)
                messages.success(request, 'Your password has been updated')
                return redirect (reverse('customer:profile'))

        elif request.POST.get('action') == 'update_phone':
            # get firebase user data
            firebase_user = auth.verify_id_token(request.POST.get('id_token'))

            request.user.customer.phone_number = firebase_user['phone_nuumber']
            request.user.customer.save()
            return redirect(reverse('customer:profile'))

    return render(request, 'customer/profile.html', {
        "user_form": user_form,
        "customer_form": customer_form,
        "password_form": password_form
    })

@login_required(login_url='/sign-in/?next=/customer/')
def payment_method_page(request):
    current_customer = request.user.customer

    #Remove existing card
    if request.method == "POST":
        stripe.PaymentMethod.detach(current_customer.stripe_payment_method_id)
        current_customer.stripe_payment_method_id = ""
        current_customer.stripe_card_last4 = ""
        current_customer.save()
        return redirect(reverse('customer:payment_method'))

    #Save stripe customer infor
    if not current_customer.stripe_customer_id:
        customer=stripe.Customer.create()
        current_customer.stripe_customer_id=customer['id']
        current_customer.save()

    # Get stripe payment method
    stripe_payment_methods = stripe.PaymentMethod.list(
        customer = current_customer.stripe_customer_id,
        type = "card",
    )

    print(stripe_payment_methods)

    if stripe_payment_methods and len(stripe_payment_methods.data) > 0:
        payment_method = stripe_payment_methods.data[0]
        current_customer.stripe_payment_method_id = payment_method.id
        current_customer .stripe_card_last4 = payment_method.card.last4
        current_customer.save()

    else:
        current_customer.stripe_payment_method_id = ""
        current_customer .stripe_card_last4 = ""
        current_customer.save()

    if not current_customer.stripe_payment_method_id:

        intent=stripe.SetupIntent.create(
            customer=current_customer.stripe_customer_id
    )
        return render (request,'customer/payment_method.html',{
            "client_secret":intent.client_secret,
            "STRIPE_API_PUBLIC_KEY":settings.STRIPE_API_PUBLIC_KEY,
    })
    else:
        return render(request, 'customer/payment_method.html')

@login_required(login_url="/sign-in/?next=/customer/")
def create_job_page(request):

    current_customer = request.user.customer
    if not request.user.customer.stripe_payment_method_id:
        return redirect(reverse('customer:payment_method'))

    has_current_job = Job.objects.filter(
        customer = current_customer,
        status__in = [
            Job.PROCESSING_STATUS,
            Job.PICKING_STATUS,
            Job.DELIVERING_STATUS
        ]
    ).exists()

    if has_current_job:
        messages.warning(request,'You currently have a processing job.')
        return redirect(reverse("customer:current_jobs"))

    creating_job = Job.objects.filter(customer=current_customer, status =
Job.CREATING_STATUS).last()
    step1_form = forms.JobCreatedStep1Form(instance=creating_job)
    step2_form = forms.JobCreatedStep2Form(instance=creating_job)
    step3_form = forms.JobCreatedStep3Form(instance=creating_job)


    if request.method == "POST":
        if request.POST.get('step') == '1':
            step1_form = forms.JobCreatedStep1Form(request.POST, request.FILES,
instance=creating_job)
            if step1_form.is_valid:
                creating_job = step1_form.save(commit=False)
                creating_job.customer = current_customer
                creating_job.save()
                return redirect(reverse('customer:create_job'))

        elif request.POST.get('step')=='2':
            step2_form =
forms.JobCreatedStep2Form(request.POST,instance=creating_job)
            if step2_form.is_valid():
                creating_job=step2_form.save()
                return redirect(reverse('customer:create_job'))

        elif request.POST.get('step')=='3':
            step3_form =
forms.JobCreatedStep3Form(request.POST,instance=creating_job)
            if step3_form.is_valid():
                creating_job = step3_form.save()
                try:
                    r =
requests.get("https://maps.googleapis.com/maps/api/distancematrix/json?origin={}&dest
inations={}&mode=transit&key={}".format(
                    creating_job.pickup_address,
                    creating_job.delivery_address,
                    settings.GOOGLE_MAP_API_KEY,
                ))
```

```
                    print(r.json()['rows'])
                    distance =
r.json()['rows'][0]['elements'][0]['distance']['value']
                    duration =
r.json()['rows'][0]['elements'][0]['duration']['value']
                    creating_job.distance = round(distance / 1000, 2)
                    creating_job.duration = int(duration/ 60)
                    creating_job.price = creating_job.distance * 20 # 20rs per km
                    creating_job.save()
                except Exception as e:
                    print(e)
                    messages.error(request, "Unfortunately we did not support
shipping at this address")
                #creating_job=step3_form.save()
                return redirect(reverse('customer:create_job'))

        elif request.POST.get('step') =='4':
            if creating_job.price:
                try:
                    payment_intent = stripe.PaymentIntent.create(
                        amount=int(creating_job.price *100),
                        currency='inr',
                        customer=current_customer.stripe_customer_id,
                        payment_method=current_customer.stripe_payment_method_id,
                        off_session=True,
                        confirm=True,
                    )

                    Transaction.objects.create(
                        stripe_payment_intent_id = payment_intent['id'],
                        job = creating_job,
                        amount = creating_job.stripe.price
                    )

                    creating_job.status = Job.PROCESSING_STATUS
                    creating_job.save()

                    #send push notification to all couriers

                    couriers = Courier.objects.all()
                    registration_tokens = [i.fcm_token for i in couriers if
i.fcm_token]
                    message=messaging.MulticastMessage(
                        notification=messaging.Notification(
                            title=creating_job.name,
                            body = creating_job.discription,
                        ),
                        webpush=messaging.WebpushConfig(
                            notification= messaging.WebpushNotification(
                                icon = creating_job.photo.url,
                            ),
                            fcm_options=messaging.webpushFCMoptions(
                                link = settings.NOTIFICATION_URL + reverse('courier:
available_jobs'),
                            ),
                        ),
                        tokens=registration_tokens
                    )

                    response = messaging.send_multicast(message)
                    print('{0} messages were sent
successfully'.format(response.success_count))

                    return redirect(reverse('customer:home'))

                except stripe.error.CardError as e:
                    err = e.error
                    # Error code will be authentication_required if authentication is
needed
                    print("Code is: %s" % err.code)
                    payment_intent_id = err.payment_intent['id']
                    payment_intent = stripe.PaymentIntent.retrieve(payment_intent_id)

     #Determnine the current step

    if not creating_job:
        current_step = 1
    elif creating_job.delivery_name:
        current_step = 4
    elif creating_job.pickup_name:
        current_step = 3
    else:
        current_step = 2

    return render(request, 'customer/create_job.html',{
        "step1_form": step1_form,
        "step2_form": step2_form,
        "step3_form": step3_form,
        "job": creating_job,
        "step": current_step,
        "GOOGLE_MAP_API_KEY": settings.GOOGLE_MAP_API_KEY
    })
```

```
@login_required(login_url="/sign-in/?next=/customer/")
def current_jobs_page(request):
    jobs = Job.objects.filter(
        customer = request.user.customer,
        status__in=[
            Job.PROCESSING_STATUS,
            Job.PICKING_STATUS,
            Job.DELIVERING_STATUS
        ]
    )
    return render(request, 'customer/jobs.html', {
        "jobs":jobs
    })

@login_required(login_url="/sign-in/?next=/customer/")
def archived_jobs_page(request):
    jobs = Job.objects.filter(
        customer = request.user.customer,
        status__in=[
            Job.COMPLETED_STATUS,
            Job.CANCELED_STATUS
        ]
    )
    return render(request, 'customer/jobs.html',{
        "jobs":jobs
    })

@login_required(login_url="/sign-in/?next=/customer/")
def job_page(request, job_id):
    job = Job.objects.get(id=job_id)

    if request.method == "POST" and job.status == job.PROCESSING_STATUS:
        job.status = job.CANCELED_STATUS
        job.save()
        return redirect(reverse('customer:archived_jobs'))

    return render (request, 'customer/job.html',{
        "job": job,
        "GOOGLE_MAP_API_KEY": settings.GOOGLE_MAP_API_KEY
    })
```

Courier Class:

Apis.py:

```
from django.http import JsonResponse
from asgiref.sync import async_to_sync
from channels.layers import get_channel_layer
from django.contrib.auth.decorators import login_required
from django.views.decorators.csrf import csrf_exempt
from core.models import *
from django.utils import timezone

@csrf_exempt
@login_required(login_url="/courier/sign-in/")
def available_jobs_api(request):
    jobs = list(Job.objects.filter(status=Job.PROCESSING_STATUS).values())

    return JsonResponse({
        "success":True,
        "jobs": jobs
    })

@csrf_exempt
@login_required(login_url="/courier/sign-in/")
def current_job_update_api(request, id):
    job = Job.objects.filter(
        id=id,
        courier=request.user.courier,
        status__in=[
            Job.PICKING_STATUS,
            Job.DELIVERING_STATUS
        ]
    ).last()

    if job.status == Job.PICKING_STATUS:
        job.pickup_photo = request.FILES['pickup_photo']
        job.pickedup_at = timezone.now()
        job.status = Job.DELIVERING_STATUS
        job.save()

        try:
            layer=get_channel_layer()
            async_to_sync(layer.group_send)("job_"+str(job.id),{
                'type':'job_update',
                'job':{
                    'status':job.get_status_display(),
                    'pickup_photo':job.pickup_photo.url,
                }
            })
        except:
            pass

    elif job.status == Job.DELIVERING_STATUS:
        job.delivery_photo = request.FILES['delivery_photo']
        job.delivered_at = timezone.now()
        job.status = Job.COMPLETED_STATUS
        job.save()

        try:
            layer=get_channel_layer()
            async_to_sync(layer.group_send)("job_"+str(job.id),{
                'type':'job_update',
                'job':{
                    'status':job.get_status_display(),
                    'delivery_photo':job.delivery_photo.url,
                }
            })
        except:
            pass


        return JsonResponse({
            "success":True
        })

@csrf_exempt
@login_required(login_url="/courier/sign-in/")
def fcm_token_update_api(request):
    request.user.courier.fcm_token = request.GET.get('fcm-token')
    request.user.courier.save()

    return JsonResponse({
        "success": True
    })
```

Forms.py:

```
from django import forms
from core.models import Courier
class PayoutForm(forms.ModelForm):
    class Meta:
        model=Courier
        fields = ('paypal_email',)
```

Views.py:

```
from asgiref.sync import async_to_sync
from channels.layers import get_channel_layer
from django.contrib import messages
from core.courier import forms
from django.shortcuts import redirect, render
from django.contrib.auth.decorators import login_required
from django.urls import reverse
from django.conf import settings

from core.models import *

# Create your views here.

@login_required(login_url="/sign-in/?next=/courier/")
def home(request):
    return redirect(reverse('courier:available_jobs'))

@login_required(login_url="/sign-in/?next=/courier/")
def available_jobs_page(request):
    return render (request, 'courier/available_jobs.html',{
        "GOOGLE_MAP_API_KEY": settings.GOOGLE_MAP_API_KEY
    })

@login_required(login_url="/sign-in/?next=/courier/")
def available_job_page(request, id):
    job = Job.objects.filter(id=id, status = Job.PROCESSING_STATUS).last()

    if not job:
        return redirect(reverse('courier:available_jobs'))

    if request.method == 'POST':
        job.courier = request.user.courier
        job.status = job.PICKING_STATUS
        job.save()

        try:
            layer=get_channel_layer()
            async_to_sync(layer.group_send)("job_"+str(job.id),{
                'type':'job_update',
                'job':{
                    'status':job.get_status_display(),
                    'delivery_photo':job.delivery_photo.url,
                    }
                })
            except:
                pass

            return redirect(reverse('courier:available_jobs'))

    return render (request, 'courier/available_job.html',{
        "job":job
    })

@login_required(login_url="/sign-in/?next=/courier/")
def current_job_page(request):
    job = Job.objects.filter(
        courier = request.user.courier,
        status__in=[
            Job.PICKING_STATUS,
            Job.DELIVERING_STATUS
        ]
    ).last()
    return render(request,'courier/current_job.html',  {
        "job":job,
        "GOOGLE_MAP_API_KEY":settings.GOOGLE_MAP_API_KEY
    })

@login_required(login_url="/sign-in/?next=/courier/")
def current_job_take_photo_page(request):
    job = Job.objects.filter(
        id = id,
        courir=request.user.courier,
        status__in=[
            Job.PICKING_STATUS,
            Job.DELIVERING_STATUS
        ]
    ).last()

    if not job:
        return redirect(reverse('courier:current_job'))
    return render(request,'courier/current_job_take_photo.html',{
    "job":job
    })

def job_complete_page(request):
    return render (request, 'courier/job_complete.html')
```

```python
@login_required(login_url="/sign-in/?next=/courier/")
def archived_jobs_page(request):
    jobs = Job.objects.filter(
        courier =  request.user.courier,
        status=Job.COMPLETED_STATUS
    )
    return render(request,'courier/archived_jobs.html',{
    "jobs":jobs
})


@login_required(login_url="/sign-in/?next=/courier/")
def profile_page(request):
    jobs = Job.objects.filter(
        courier = request.user.courier,
        status=Job.COMPLETED_STATUS
    )
    total_earnings=round(sum(job.price for job in jobs)*0.8,2)
    total_jobs = len(jobs)
    total_km=sum(job.distance for job in jobs)
    return render(request,'courier/profile.html',{
        "total_earnings":total_earnings,
        "total_jobs":total_jobs,
        "total_km":total_km
    })

@login_required(login_url="/sign-in/?next=/courier/")
def payout_method_page(request):
    payout_form=forms.PayoutForm(instance=request.user.courier)
    if request.method =='POST':
        payout_form=forms.PayoutForm(request.POST,instance=request.user.courier)
        if payout_form.is_valid():
            payout_form.save()
            messages.success(request,"Payout address is updated.")
            return redirect(reverse('courier:profile'))
    return render(request,'courier/payout_method.html',{
        "payout_form": payout_form
    })
```

Customer Sign-in.html:

```
{% extends 'base.html' %}
{% load bootstrap4 %}
{% block content %}

<div class="container-fluid mt-5">
    <div class="row justify-content-center">
        <div class="col-lg-4">
            <div class="card">
                <div class="card-body">
                    <h4 class="text-center text-uppercase mb-3">
                        <b>
                            {% if request.GET.next != '/courier/' %}
                            Customer
                            {% else %}
                            Courier
                            {% endif %}
                        </b>
                    </h4>

                    <form method="POST">
                        {% csrf_token %}
                        {% bootstrap_form_errors form %}
                        {%bootstrap_label "Email" %}
                        {% bootstrap_field form.username show_label=false
placeholder="Email" %}
                        {% bootstrap_field form.password %}
                        <button class="btn btn-warning btn-block">Sign In</button>
                        <p class="text-center mt=3">
                            New to Drop Ship? <a href="/sign-up/?next={{
request.GET.next }}"><b>Sign Up</b></a>
                        </p>
                        <hr>
                        <a href="{% url 'social:begin' 'facebook' %}?next={{
request.GET.next }}"
                        class="btn btn-outline-primary btn-block">Sign In with
Facebook</a>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>

{% endblock %}
```

Customer Sign-up.html:

```
{% extends 'base.html' %}
{% load bootstrap4 %}

{% block content %}

<div class="container-fluid mt-5">
    <div class="row justify-content-center">
        <div class="col-lg-4">
            <div class="card">
                <div class="card-body">
                    <h4 class="text-center text-uppercase mb-3">
                        <b>
                            {% if request.GET.next != '/courier/' %}
                            Customer
                            {% else %}
                            Courier
                            {% endif %}
                        </b>
                    </h4>

                    <form method="POST">
                        {% csrf_token %}
                        {% bootstrap_form form %}
                        <button type="submit" classs="btn btn-warning btn-block">Sign
Up</button>
                        <p class="text-center mt=3">
                            Already have an account? <a href="/sign-in/?next={{
request.GET.next }} "><b>Sign In</b></a>
                        </p>
                        <hr>
                        <a href="{% url 'social:begin' 'facebook' %}?next={{
request.GET.next }}"
                        class="btn btn-outline-primary btn-block">Sign In with
Facebook</a>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>

{% endblock %}
```

Load-firebase.html:

```
<script type="module">
    // Import the functions you need from the SDKs you need
    import { initializeApp } from "https://www.gstatic.com/firebasejs/9.8.1/firebase-
app.js";
    import { initializeApp } from "https://www.gstatic.com/firebasejs/9.8.1/firebase-
auth.js";
    import { initializeApp } from "https://www.gstatic.com/firebasejs/9.8.1/firebase-
messaging.js";
    // TODO: Add SDKs for Firebase products that you want to use
    // https://firebase.google.com/docs/web/setup#available-libraries

    // Your web app's Firebase configuration
    const firebaseConfig = {
      apiKey: "AIzaSyB7Cm3LgiY5KOpy8tgRB8SE9vE6Kc8fETo",
      authDomain: "dropship-f1ad9.firebaseapp.com",
      projectId: "dropship-f1ad9",
      storageBucket: "dropship-f1ad9.appspot.com",
      messagingSenderId: "161989651929",
      appId: "1:161989651929:web:fad9968a82ac7c7cd7fdc8"
    };

    // Initialize Firebase
    const app = initializeApp(firebaseConfig);
  </script>
```

Create-job.html:

```
{% extends "base.html" %}
{% load bootstrap4  %}
{% block head %}

<script
    src="https://maps.googleapis.com/maps/api/js?key={{ GOOGLE_MAP_API_KEY
}}&callback=initMap&libraries=places&v=weekly"
    defer
></script>

<style>
    #pills-tab a {
        color: black;
    }

    #pills-tab a:hover{
        color: orange;
    }

    #pills-tab a.active{
        color: orange;
    }
    #pickup-map #delivery-map{
        height: 100%;
    }

</style>
{% endblock %}
{% block content %}
<div class="container mt-4">
    <div class="row">
        <!-- LEFT SIDE -->
        <div class="col-lg-4">
            <div class="card">
                <div class="card-header">
                    JOB SUMMARY
                </div>
                <div class="card-body">
                    {% if not job %}
                    <p>A summary of your job information will appear here</p>
                    {% else %}

                    {% if step > 1 %}
                    <h4>{{ job.name }}</h4>
                    <span>{{ job.quantity }} Item</span>
                    <span>{{ job.get_size_display }} Job</span>
                    {% endif %}

                    {% if step > 2 %}
                    <hr/>
                    <p class="text-secondary"><small><b>PICKUP</b></small></p>
                    <h4>{{ job.pickup_name }}</h4>
                    <span>{{ job.pickup_address }}</span>
                    {% endif %}

                    {% if step > 3 %}
                    <hr/>
                    <p class="text-secondary"><small><b>DELIVERY</b></small></p>
                    <h4>{{ job.delivery_name }}</h4>
                    <span>{{ job.delivery_address }}</span>
                    {% endif %}
                    {% endif %}
                </div>
            </div>
        </div>
        <!-- RIGHT SIDE -->
        <div class="col-lg-8">

            <!--step tabs-->
            <div class="card mb-5">
                <div class="card-body">
                    <ul class="nav nav-pills nav-justified align-items-center mb-3"
id="pills-tab" role="tablist">
                        <li class="nav-item" role="presentation">
                          <a class="{% if step == 1 %}active{% endif %}" id="pills-
info-tab" data-toggle="pill" href="#pills-info" role="tab" aria-controls="pills-info"
aria-selected="true">Item Info</a>
                        </li>
                        <li class="nav-item" role="presentation">
                          <a class="{% if step == 2 %}active{% endif %}" id="pills-
pickup-tab" data-toggle="pill" href="#pills-pickup" role="tab" aria-controls="pills-
pickup" aria-selected="false">Pickup</a>
                        </li>
                        <li class="nav-item" role="presentation">
                          <a class="{% if step == 3 %}active{% endif %}" id="pills-
delivery-tab" data-toggle="pill" href="#pills-delivery" role="tab" aria-
controls="pills-delivery" aria-selected="false">Delivery</a>
                        </li>
                        <li class="nav-item" role="presentation">
                            <a class="{% if step == 4 %}active{% endif %}" id="pills-
payment-tab" data-toggle="pill" href="#pills-payment" role="tab" aria-
controls="pills-payment" aria-selected="false">Payment</a>
                        </li>
                      </ul>
                </div>
            </div>

            <!--Step forms-->
            <b>CREATE A JOB</b>
            <div class="tab-content" id="pills-tabContent">

                <!--step 1-->
                <div class="tab-pane fade {% if step == 1 %}show active{% endif %}"
id="pills-info" role="tabpanel" aria-labelledby="pills-info-tab">
                    <h1>Item Info</h1>

                    <form method="POST" enctype="multipart/form-data">
```

```
                        <b class="text-secondary">Item Information</b><br/>
                        <div class="card bg-white mt-2 mb-5">
                            <div class="card-body">
                                {% csrf_token %}
                                {% bootstrap_form step1_form %}
                            </div>
                        </div>
                        <input type="hidden" name="step" value="1">
                        <button type="submit" class="btn btn-warning">Save &
Continue</button>
                    </form>
                </div>

                <!--step 2-->
                <div class="tab-pane fade {% if step == 2 %}show active{% endif %}"
id="pills-pickup" role="tabpanel" aria-labelledby="pills-pickup-tab">
                    <h1>Pickup</h1>
                    <form method="POST" enctype="multipart/form-data">
                        <b class="text-secondary">Pickup Information</b><br/>
                        <div class="card bg-white mt-2 mb-5">
                            <div class="card-body">

                                <div class="row">
                                    <div class="col-lg-8">
                                        {% csrf_token %}
                                        {% bootstrap_form step2_form
exclude='pickup_lat, pickup_lng' %}
                                        <input hidden id="pickup_lat"
name="pickup_lat" value= "{{ job.pickup_lat }}">
                                        <input hidden id="pickup_lng"
name="pickup_lng" value= "{{ job.pickup_lng }}">
                                    </div>
                                    <div class="co-lg-4">
                                        <div id="pickup-map"></div>
                                        <div id="pickup-infowindow-content">
                                            <img src="" width="16" height="16"
id="pickup-place-icon"/>
                                            <span id="pickup-place-name"
class="title"></span> <br/>
                                            <span id="pickup-place-address"></span>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                        <input type="hidden" name="step" value="2">
                        <button type="button" class="btn btn-outline-warning"
                        onclick="$('#pills-info-tab').tab('show');">Back</button>
                        <button type="submit" class="btn btn-warning">Save &
Continue</button>
                    </form>
                </div>

                <!--step 3-->
                <div class="tab-pane fade {% if step == 3 %}show active{% endif %}"
id="pills-delivery" role="tabpanel" aria-labelledby="pills-delivery-tab">
                    <h1>Delivery</h1>
                    <form method="POST" enctype="multipart/form-data">
                        <b class="text-secondary">Delivery Information</b><br/>
                        <div class="card bg-white mt-2 mb-5">
                            <div class="card-body">

                                <div class="row">
                                    <div class="col-lg-8">
                                        {% csrf_token %}
                                        {% bootstrap_form step3_form
exclude='delivery_lat, delivery_lng' %}
                                        <input hidden id="delivery_lat"
name="delivery_lat" value= "{{ job.delivery_lat }}">
                                        <input hidden id="delivery_lng"
name="delivery_lng" value= "{{ job.delivery_lng }}">
                                    </div>
                                    <div class="co-lg-4">
                                        <div id="delivery-map"></div>
                                        <div id="delivery-infowindow-content">
                                            <img src="" width="16" height="16"
id="delivery-place-icon"/>
                                            <span id="delivery-place-name"
class="title"></span> <br/>
                                            <span id="delivery-place-address"></span>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                        <input type="hidden" name="step" value="3">
                        <button type="button" class="btn btn-outline-warning"
                        onclick="$('#pills-info-tab').tab('show');">Back</button>
                        <button type="submit" class="btn btn-warning">Save &
Continue</button>
                    </form>
                </div>
                <!--step 4-->
                <div class="tab-pane fade {% if step == 4 %}show active{% endif %}"
id="pills-payment" role="tabpanel" aria-labelledby="pills-payment-tab">
                    <h1>Payment</h1>

                    <form method="POST">
                        <b class="text-secondary">Payment Method</b>
                        <div class="card bg-white mt-2 mb-5">
                            <div class="card-body">
                                {% csrf_token %}
                                <div class="form-group'>
                                    <label>Your Credit/Debit Card</label>
                                    <input class="form-control" value="**** **** ****
{{ request.user.customer.stripe_card_last4 }}" disabled>
                                </div>
                                <div class="form-group">
                                    <label>Price</label>
                                    <input class="form-control" value="${{ job.price
}}" disabled>
                                </div>
                            </div>
                        </div>
                        <input type="hidden" name="step" value="4">
                            <button type="button" class="btn btn-outline-warning"
                                onclick="$('#pills-delivery-
tab').tab('show');">Back</button>
                            <button type="submit" class="btn btn-warning">Create
Job</button>
                     </form>
                </div>
            </div>
        </div>
    </div>
</div>

<script>
    var pickupLat = parseFloat('{{ job.pickup_lat }}');
    var pickupLng = parseFloat('{{ job.pickup_lng }}');

    var deliveryLat = parseFloat('{{ job.delivery_lat }}');
    var deliiveryLng = parseFloat('{{ job.delivery_lng }}');

    function initMapByType(type, initLat, initLng){
        const map = new google.maps.Map(document.getElementById(type + "-map"),{
            center: { lat: initLat || 40.749933, lng: initLng || -73.98633 },
            zoom: 13,
        });

        if(initLat && initLng){
            new google.maps.Marker({
                position:new google.maps.LatLng(initLat,initLng),
                map:map,
            })
        }

        const input = document.getElementById("id_"+ type + "_address");
        const autocomplete = new google.maps.places.Autocomplete(input);
        autocomplete.bindTo("bounds", map);
        autocomplete.setFields(["address_components", "geometry", "icon", "name"]);
        const infowindow = new google.maps.InfoWindow();
        const infowindowContent = document.getElementById(type +" -infowindow-
content");
        infowindow.setContent(infowindowContent);
        const marker = new google.maps.Marker({
            map,
            anchorPoint: new google.maps.Point(0, -29),
        });
        autocomplete.addListener("place_changed", () => {
        infowindow.close();
        marker.setVisible(false);
        const place = autocomplete.getPlace();
        if (!place.geometry) {
            // User entered the name of a Place that was not suggested and
            // pressed the Enter key, or the Place Details request failed.
            window.alert("No details available for input: '" + place.name + "'");
        return;
        }
        if (place.geometry.viewport) {
            map.fitBounds(place.geometry.viewport);
        } else {
            map.setCenter(place.geometry.location);
            map.setZoom(17);
        }
        marker.setPosition(place.geometry.location);
        marker.setVisible(true);
        let address = "";
        if(place.address_components){
            address=[
                (place.address_components[0] &&
                    place.address_components[0].short_name)||
                    "",
                (place.address_components[1]&&
                    place.address_components[1].short_name)||
                    "",
                (place.address_components[2]&&
                    place.address_components[2].short_name)||
                    "",
            ].join(" ");
        }
        infowindowContent.children[type +"-place-icon"].src=place.icon;
        infowindowContent.children[type +"-place-name"].textContent=place.name;
        infowindowContent.children[type +"-place-address"].textContent=address;
        infowindow.open(map,marker);

        $("#"+ type +"_lat").val(place.geometry.location.lat());
        $("#"+ type +"_1ng").val(place.geometry.location.lng());
    });I
}

    function initMap(){
        initMapByType("pickup", pickupLat, pickupLng);
        initMapByType("delivery", deliveryLat, deliiveryLng);
    }
</script>
{% endblock %}
```