Courierdjango - courier management service using django webframe

CSIT (Tribhuvan Vishwavidalaya)

Soch College of IT

Affiliated to

Tribhuvan University

Institute of Science and Technology

Project Report on

Courier Management System

Submitted to

Soch College of IT

Ranipauwa, Pokhara

In partial fulfillment of the requirement for the Bachelor Degree in Computer Science and Information Technology

Submitted by:

Arvind Karki (16835/074)
Ashish Baniya (16836/074)
Susham Malla (16863/074)

February, 2022

SUPERVISOR'S RECOMMENDATION

The project "Courier Management System" submitted by Arvind Karki (16835/074), Ashish Baniya (16836/074) and Susham Malla (16863/074) of Soch college of IT, Ranipauwa, Pokhara-11, is prepared under the supervision as per the procedure and format requirements laid by the Faculty of Science and Technology, Tribhuvan University, as partial fulfillment of the requirements for the award of the degree of Bachelor of Science in Computer Science and Information Technology (B.Sc. CSIT). I, therefore, recommend the project work report for evaluation.

……

Mr. Anil Thapa

Lecturer

Soch college of IT

Date:

LETTER OF APPROVAL

We hereby endorse the project work entitled "Courier Management System" submitted by Arvind Karki (16835/074), Ashish Baniya (16836/074) and Susham Malla (16863/074) of Soch college of IT, Ranipauwa, Pokhara-11, in partial fulfillment of the requirements for the award of degree of Bachelor of Science in Computer Science and Information Technology (B.Sc. CSIT) for the external evaluation. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

Signature of Supervisor	Signature of HOD/Coordinator
……	……
Mr. Anil Thapa	Mr. Sushil Adhikari
Lecturer,	Principal,
Soch college of IT,	Soch college of IT,
Ranipauwa, Pokhara-11	Ranipauwa, Pokhara-11
Signature of Internal Examiner	Signature of External Examiner
……	……
Mr. Raghu Nath Gyawali	Er. Hari Prasad Baral
Lecturer	Asst. Professor
Soch College of IT,	Institute of Engineering,
Ranipauwa, Pokhara-11	Paschimanchal Campus, Pokhara

Date of approval:

DECLARATION

We hereby declare that the project work entitled “Courier Management System” submitted to the Faculty of Science and Technology, Tribhuvan University, Kathmandu is an original piece of work under the supervision of Mr. Anil Thapa faculty members, Soch college of IT, Ranipauwa, Pokhara-11 and is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Science in Computer Science and Information Technology (B.Sc. CSIT). This project work report has not been submitted to any other university or institution for the award of any degree or diploma.

Submitted By:

Arvind Karki (16835/074)

Ashish Baniya (16836/074)

Susham Malla (16863/074)

B.Sc. CSIT, 2074 Batch

Soch College of IT, Pokhara

ACKNOWLEDGEMENT

First of all, we would like to thank Tribhuvan University and our college Principal and Vice Principal Mr. Sushil Adhikari and Mr. Bikash Pahari for providing us an opportunity to apply academic knowledge under supervised hands of field conditions. We would like to express our sincere gratitude to our subject project head and project supervisor Mr. Anil Thapa. We really appreciate the help and support you have provided us during the course of the project work. We are also thankful to all members of department for helping us during our entire project. Finally, we wish to extend our sincere gratitude to all our respective teachers of Soch College of IT for giving us guidelines to prepare this project and thank you to all our classmates for their cooperation and encouragement to help us develop this project.

ABSTRACT

People nowadays place a high value on sending and receiving products such as imported furniture, electronic devices, gifts, commercial goods, and the like. People rely heavily on various modes of transportation, the majority of which rely on manual methods of receiving and delivering goods. There is no method to trace the products until they are delivered, and there is no means to inform the consumer about what happened during transportation after he ordered some. In such a case, we require a system that fully automates shipping activities, including customer manage their courier, check status and interact with management. This requirement is met by Courier Management System software, which is an online application for cargo management professionals that allows them to receive items from a source, deliver them to a specified destination, and watch their progress over time.

Contents

List of figures:

Screenshots:

List of tables:

Chapter 1: Introduction

1.1 Introduction:

The main concern of the courier agency is to deliver the right parcel to its owner. To accomplish it the agency has to maintain the record of the customer details, courier, bill, payment, courier tracking ID. In order to maintain the manual work and to create a reliable working platform, computerized courier management system is an essential component in all courier agencies these days.

The logistics courier management system project, built with the Django framework, is an online application designed to automate the many processes in the load office. It focuses on managing and implementing a courier agency's exercises and strategies. This program aids in the reduction of human effort and errors caused by traditional working practices. It will keep track of all data and display the courier's and customers' descriptions. Hence, speed up the delivery process and increases the customer satisfaction, and where it has ability for receiver to accept and reject the courier.

It is responsible for scheduling order deliveries, assigning them to couriers, overseeing operations while couriers are in the field, and suggesting strategic improvements for future planning. Depending on the size and complexity of a business, courier management system may also encompass certain aspects of vehicle oversight, fuel management, and courier expense approval.

The courier service is one of the solutions of these problems. It is used to mail items to anyone in the world in a timely manner.

1.2 Problem Statement:

When people transfer their products via any courier service, they want to know whether their products has been delivered to the correct location or not, and if not, when it will be delivered and where it is currently. Manually collecting all of this data is a complex and time-consuming task. Various processes and paper work from the management side are also required to manage all of these activities.

1.3 Objectives:

This project deals with the 'Courier management'. The system is used for daily activities such as booking, loading, delivery, status check, and managing branches. It is very difficult to do this process manually. Hence it is recommended to computerize the process by developing the relative software as the world is turning into information and technology; computerization becomes necessity in all walks of life.

The Objective of Courier Management Systems is:

- Maximize work capacity by supporting couriers
- To automate workflow
- To consistently meet of surpass due to dates and time and to keep data records

1.4 Scope and limitation:

Courier management computerization is "the incorporate of appropriate technology to help administrator manage information. Technology is considered appropriate, when it utilizes the most abundant domestic resources and conserves capital and skilled personnel". This project deals with the maintenance of booking details, incoming courier details, courier non delivery details and courier return details etc. The main aim of this project is to computerize the maintenance of courier management.

Limitation:

- Strict limitation in package dimension and weight
- Daily transactions are to be entering into different books immediately to avoid conflicts which are very difficult.
- More manual hours need to generate required reports.

1.5 Methodology:

Because the project's planning and requirements have not been thoroughly discussed, the methodology to be employed for this project is agile. The Agile methodology is a style of project management that divides a project into phases. It necessitates ongoing engagement with stakeholders as well as continual development at each stage. Team's cycle through a process

of planning, executing, and assessing once the job begins. Collaboration is essential among team members as well as project stakeholders.

Figure 1:agile methodology (source: https://www.nvisia.com/insights/agile-methodology)

1.6 Report Organization:

This study has been organized into following six chapters:

In chapter 1: we introduce why our system is built i.e., the problem definition, objectives of the project and its features along with its scope and limitation.

In chapter 2: we review the existing literature that contains reviews of journals and articles, and earlier thesis related to the subject. We also discuss about the background study.

In chapter 3: we discuss about the system analysis. The system analysis includes system analysis, requirement analysis and feasibility analysis.

In chapter 4: we discuss about system design and algorithm details. Class diagram, Object diagram, Activity diagram etc. are shown.

In chapter 5: we discuss about the implementation and testing process. In this chapter we also discuss about tools that are used.

In chapter 6: we discuss about the conclusion of the project and future enhancement.

Chapter 2: Background Study and Literature Review

2.1 Background Study:

Computers have become part of the life for accessing almost any kind of information. Life in the 21st century is full of technological advancement and in this technological age it is very difficult for any organization to survive without utilizing technology. The World Wide Web contributes greatly to the creation of an ever-increasing global information database.

What we had proposed is a web-based application for record keeping of courier. The main advantage of this system is that it greatly simplifies the record keeping process for product delivery. As we know the first step of profitability is good record keeping. Records give you the information you need to make sound business management decisions.

A courier service is a business that facilitates the transportation and shipping of packages and important documents to their intended destinations. An individual courier is someone who either works as an employee of a larger company or a person who owns a sole proprietorship [1].

2.2 Literature Review:

The courier system in Nepal is mostly traditional way of paperwork and done manually. The trend of online businesses is increasing so a reliable digital way of carrying courier is a must in current scenario. There is no proper way of storing data of courier and customers. Most of the literature review was taken from documents on operation of the courier companies and some other policy books policy.[2] As a way of managing its activities manually, Courier services rely on various journals to record its transaction. The registration of personal records, all registered packages in a journal an amount paid. For example, any package that was to be delivered by the department is recorded in the package registration journal. Considering the work involved in recording these transactions; issuing receipts and generating statistical reports was time consuming. The people sometimes forget to record some data due to the pressure from the waiting queue, which may affect the accuracy of the information and the decision to be made.[3]

Due to the growing scope of 2-commerce and demand for on-time delivery, most of the company in Nepal rely on courier delivery providers in order to outsource their delivery products as soon as possible. The delivery companies assist business owner in growing their businesses nationally and internationally.[4] COVID-19 and a 120-day national lockdown (March 24th - July 22nd) plus a second 22-day lockdown in Kathmandu valley (Aug 19th - Sept 10th) caused disruptions to national and international transport systems, affecting the ability of government and humanitarian workers to respond. Online stores are confused about providing service due to lack of clear guidelines from the government regarding ecommerce. They deliver to a very limited number of places, and also only at certain times. This has made life difficult for both sellers and buyers.[5]

As courier service nature is to expand day to day, every courier service solution is a web-based on which is quite easy to access without any installation overheads. Non- web-based solutions are not available because there are not feasible and does not satisfy the purpose of expansion. With a web-based system user or branch operator can be a person who has a basic knowledge of browser and computer. But with in-house software, it will be hard to deal with installations, maintenance any kind of unexpected mees-up.[6]

We have also reviewed www.dhl.com from where DHL is a German logistics company providing courier; package delivery and express mail service, which is a division of the German logistics firm. The company group delivers over 1.6 billion parcels.[7]

Chapter 3: System Analysis

3.1 System Analysis

System analysis is the performance management and documentation of activities related to the life cycle phases of any software.

Software Analysis starts with a preliminary analysis and later switches on to a detailed one. During the preliminary analysis the Analyst takes a quick look at what is needed and whether the cost benefits. Detailed analysis studies in depth all the cornered factors, which builds and strengthens the software.

Class Diagram for Courier Management System

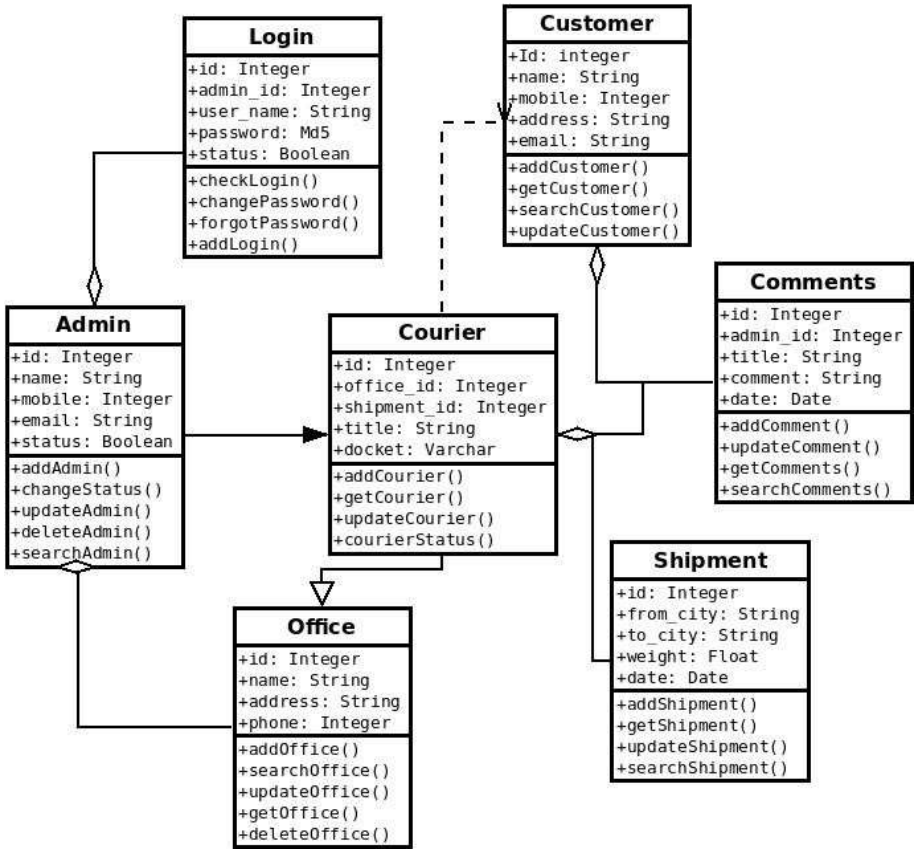


Figure 2: Class diagram of Courier Management System

A class diagram is a static diagram that is used to represent a system's static perspective. The above-mentioned diagram is the Class-Diagram of the system Hamro Books. In the following

class diagram, we have following classes: Admin, Login, Customer, Courier, Shipment, Office and Comments. Attributes are shown in the second partition of each block.

3.2 Requirement Analysis:

In the development of a system, the requirements play a critical role. After the requirements are gathered, the structure, functionality, and operational restrictions of the system are determined. Because of their dynamic and contingent nature, the needs are difficult to predict. The requirements of the system user may vary during development. Because one requirement may be dependent on another, adjustments to the lower requirement may result in changes to the upper requirements, and vice versa.

We have identified the requirements for the proposed system 'Courier Management System.'. The following are the requirements:

i. Functional requirements are those that are used to demonstrate the system's internal working nature, as well as the system's description and explanation of each subsystem. It includes determining what task the system should accomplish, the processes involved, the data the system should keep, and the user interfaces. The following are the functional requirements:

- **User's registration:** The system should allow new users to register online.
- **Send courier:** User request system to send package to desirable location.
- **Automatic update to database once sender sends courier request or new customer registered**: Data are automatically updated to database.
- **Packages:** Goods that are about to be deliver.
- **Users-** The system should allow user to view courier status.
- **Manage issues/ Complaints**—Users should allow to complain and admin should manage issue.
- **Dashboard-** Here admin can view all recent courier.
- **Sign in-** Here user can login with valid username and password.
- **Receive Courier:** Receiver with matching information will receive the courier.

Use Case diagram

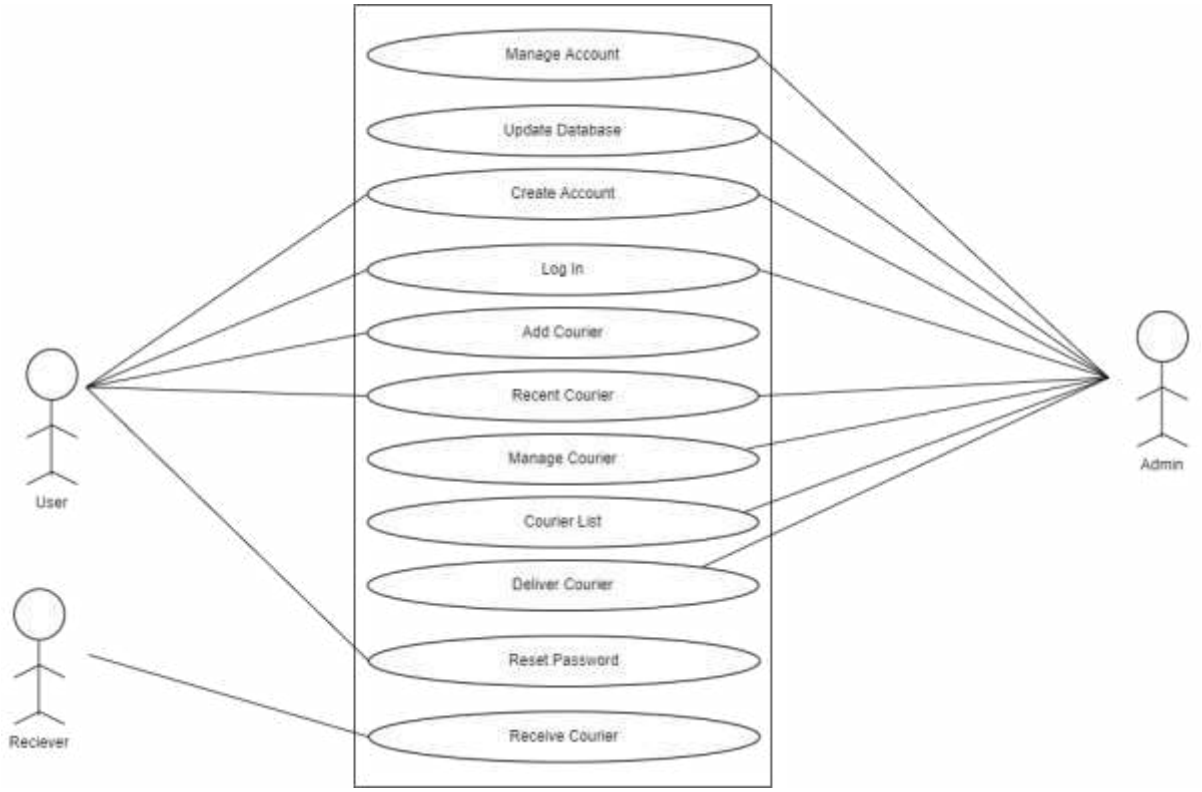


Figure 3 Use Case Diagram

Use case diagram is a behavioral UML diagram type and frequently used to analyze various systems. They enable you to visualize the different types of roles in a system and how those roles interact with the system. The above-mentioned diagram represents the use case diagram for our project Courier Management System. User is linked to its use-cases i.e., user login, add courier. Admin is linked to manage couriers and customer. Receiver is linked with receiving courier sent by sender.

ii. Non-functional requirement:

- **Easy and Simple UI:** The user interface of the application is simple and user friendly.
- **Security:** The access to the softwa0re application should restrict to the authorized users without a valid username and password.

3.3 Feasibility Study:

After doing the system study and analyzing all the existing or required functionalities of the system, the next task is to do the feasibility study for the project. All projects are feasible - given unlimited resources and infinite time

Feasibility study includes consideration of all the possible ways to provide a solution to the given problem. The proposed solution should satisfy all the user requirements and should be flexible enough so that future changes can be easily done based on the future upcoming requirement.

i. Economic Feasibility

This is an extremely crucial factor to consider when creating a project. We chose the technology with the lowest feasible cost in mind. The organization is responsible for all hardware and software costs. Overall, we estimate that the advantages the organization would gain from the proposed system will more than offset the system's initial costs and ongoing operating costs.

ii. Technical Feasibility

This included the study of function, performance and constraints that may affect the ability to achieve an acceptable system. For this feasibility study, we studied complete functionality to be provided in the system.

iii. Operational Feasibility

Last but not least, courier management software covers all aspects of operational feasibility. Customers today want user-friendly software that is understandable even by non-techies in an ever-busier world. Furthermore, new users or those who wish to become more familiar with the software and the system as a whole can participate in a guided training session to learn the basics of the software.

Chapter 4: System design

4.1 Activity Diagram

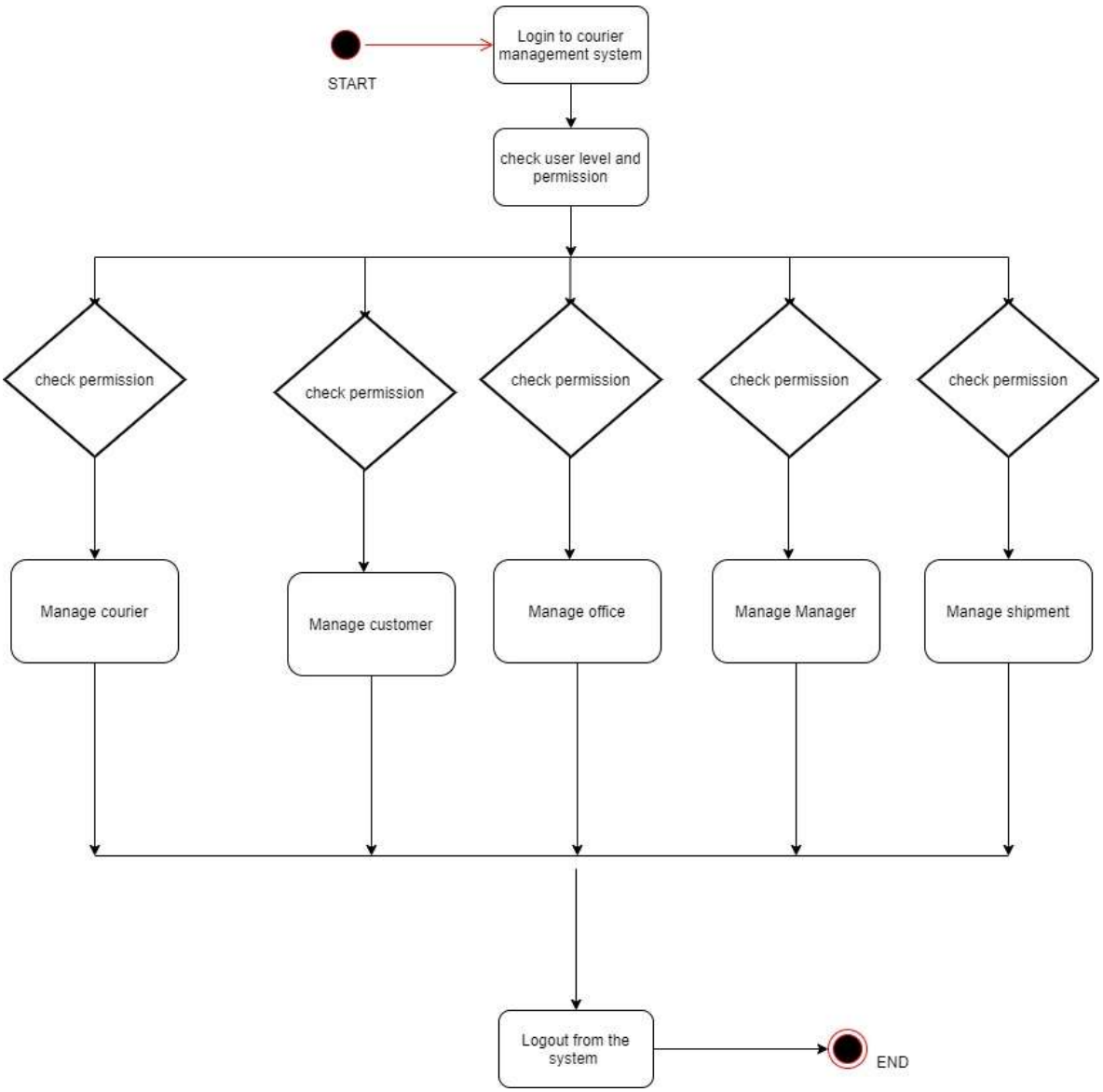


Figure 4: Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The above diagram is the State Machine Diagram of our project. Activity starts with user logging into our system then their role is given according to permissions given to them.

The users can execute activity shown in partitions below permission. Then the user logouts from the system after their work is fulfilled.

4.2 Component Diagram for Courier Management System

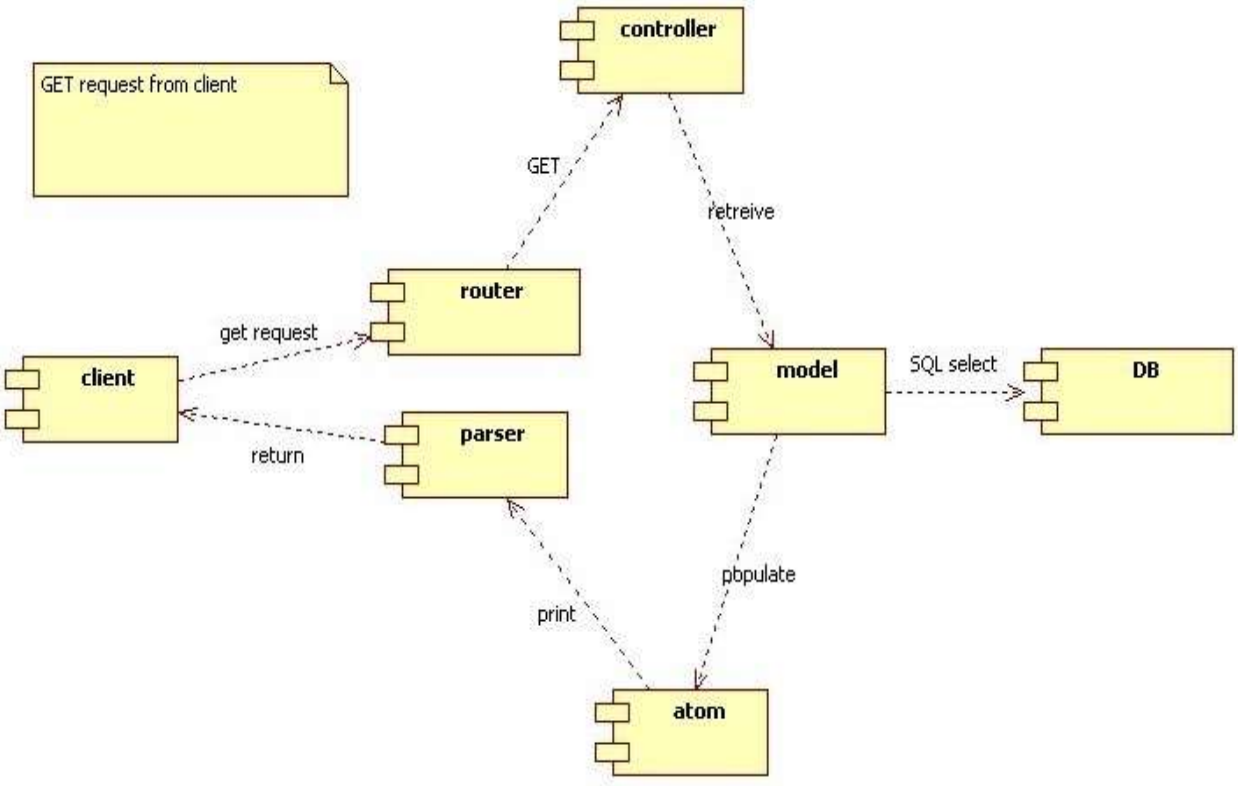


Figure 5: Component Diagram

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required functions is covered by planned development. The above figure represents component diagram of our system. Firstly, user register in the system or if user is already registered than he/she login and can access their dashboard and can use different features available in the application. They add courier request to be delivered with recipient details. After the item is picked up from the sender location, a tracking code is generated for user which will help to know about their

consignment progress report and chat with our administration for FAQs and custom questions. Then the user logs out from system.

Chapter 5: Implementation and Testing

5.1: Implementation

Implementation phase is the phase in our project development, where we implemented our conceptual design into the working program by using various tools. The successful implementation of project is nearer steps towards the project completion. Project implementation was not an easy step to us as we encountered various issues related to the programming logic as challenges. Our project is based on the Agile model. We chose Agile Software Development methodology as it is an iterative approach to the design and development of software. The Agile approach embraces the constant changes that occur in the development of technology – allowing teams to break the lengthy requirements, build, and test phases down into smaller segments, ultimately delivering working software quickly and more frequently.

5.2: Tools used

Front-End

• **HTML (Hypertext Markup Language)**: It is the standard markup language for documents to be displayed in a web browser. We used HTML to display the content and structure of the web pages in our project.

• **CSS (Cascading Style Sheet):** It describes how HTML elements are to be displayed. We used CSS to design to enable the separation of presentation and content, including layouts, colors and fonts.

• **JavaScript**: We used JavaScript for enabling interactive web pages in our project as it is the standard client-side / Frontend scripting language for web development and has no alternatives. We also used the JavaScript library, j Query to further enhance our development speed and convenience.

• **Bootstrap**: It is the framework for CSS. In our project we used bootstrap 4. We used bootstrap framework for creating layouts as it is compatible with all modern browsers.

• **jQuery**: It is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation and Ajax. In our project we used jQuery as it accomplishes and wraps many lines of JS code into a single line of code

Back-End

Python: Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. We used python in our project as it can be fully customized to match our customer needs.

Django Framework: It is web-application framework for python. Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design.

Database Management System

SQ Lite

It is an open-source database management system (DBMS). We used SQLite for handling the database in our project

Documentation Tools

- MS-Word was used as a text editor for documentation process.
- draw.io was used to draw E-R, use-case, class, activity.

Other Tools

- Text-Editor (PyCharm, Sublime)
- Web Browsers
- Git Hub – for code sharing with group member

5.3 Testing

After the project is ready, we tested its various components in terms of quality, performance to make it error free and remove any sort of technical jargons. The testing phase can be carried out manually or by using automated testing tools to ensure each component works fine.

5.3.1 Unit Testing

During the coding phase each individual module was tested to check whether it works properly or not. Different errors found during unit testing were debugged. Some of the test cases are listed below:

Table 1: Admin Login

S.N.	Test Inputs	Expected Output	Actual Output	Result
1.	Email Address:xyz@gmail.com Password: xyzxyz	Login should be successful.	Login Success	Test successful
2.	Email Address:xyz@gmail.com Password:	Login should be unsuccessful.	Required fields are empty	Test successful
3.	Email Address:xyz@gmail.com Password:abc	Login unsuccessful	Email or password doesn't match	Test successful

Table 2: Customer Login

S.N.	Test Inputs	Expected Output	Actual Output	Result
1.	Name: Hari Thapa Adders: Rambazar Contact no: 98…	Customer Login.	Customer created successful.	Test successful.
2.		Customer Login unsuccessful. All fills should be field first.	Required field should be field.	Test successful.

5.3.2 System Testing

System testing is also done after integration testing in order to ensure that the whole system functions properly. After the integration testing, the whole system working process was checked. We found that the output was as per the system specifications and hence the system was found to work properly.

Chapter 6: Conclusion and Future Recommendation

6.1 Conclusion

Courier Management System is an in-demand system toward having systematic delivery of products by various ways of transportation within a certain time and digital payment system. Traditional way of delivering product is not very effective. So accordingly, ‘Courier management’ is a web-based application being developed where proposed system would assure the accurate and productive delivery of product. The Courier Service System project provides user interface in to the application through an authenticated admin and also, it’s a modern way of usage to a Courier Service through a computer system.

6.2 Future Recommendation

In near future, we are planning to track with GPS so that customers can add their location exactly and it will keep track of the customer's courier and handle the data for the customer and digital payment system. We are planning to go automate when possible.

References

[1]"Courier - Wikipedia", *En.wikipedia.org*, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Courier. [Accessed: 02- Mar- 2022].

[2]"Nepal Customs, Trade Regulations and Procedures Handbook Volume 1 Strategic and Practical Information", *Google Books*, 2022. [Online]. Available: https://books.google.com.np/books?id=OteyDwAAQBAJ&pg=PA106&dq=courier+service+in+nepal&hl=en&sa=X&ved=2ahUKEwiNvJv_tKb2AhV2wosBHZttB44Q6AF6BAgKEAI#v=onepage&q=courier%20service%20in%20nepal&f=false. [Accessed: 02- Mar- 2022].

[3]"Courier Management System (CMS) - docshare.tips", *docshare.tips*, 2022. [Online]. Available: https://docshare.tips/courier-management-system-cms-_57503ba6b6d87f51a08b4928.html. [Accessed: 02- Mar- 2022].

[4]"Delivery companies in Nepal - Kathmandu valley to all Nepal - Pasls Blog", *Pasls Blog*, 2022. [Online]. Available: https://www.pasls.com/blog/delivery-companies-in-nepal/. [Accessed: 02- Mar- 2022].

[5]"People can't go out, and they can't get home delivery", *Kathmandupost.com*, 2022. [Online]. Available: https://kathmandupost.com/money/2021/05/14/people-can-t-go-out-and-they-can-t-get-home-delivery. [Accessed: 02- Mar- 2022].

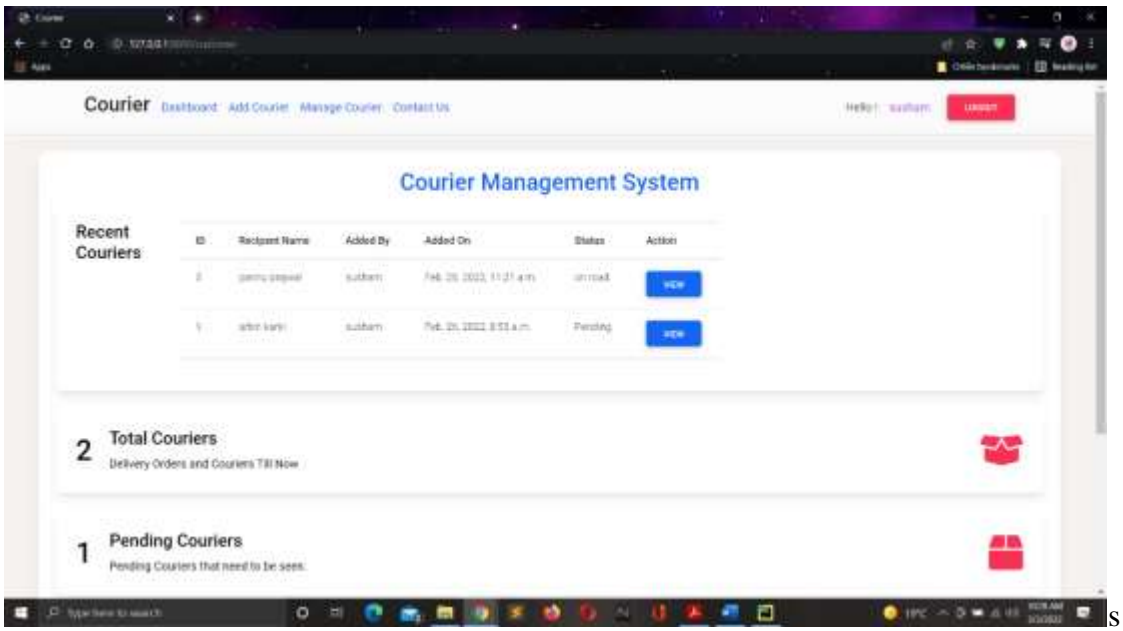
[6] *Core.ac.uk*, 2022. [Online]. Available: https://core.ac.uk/download/pdf/234675246.pdf. [Accessed: 02- Mar- 2022].

[7]"Standards and Guidelines | DHL API Developer Portal", *Developer.dhl.com*, 2022. [Online]. Available: https://developer.dhl.com/documentation/standards-guidelines. [Accessed: 02- Mar- 2022].

Appendix

Screenshot

Landing page



s

Figure 6: Landing Page

Registration

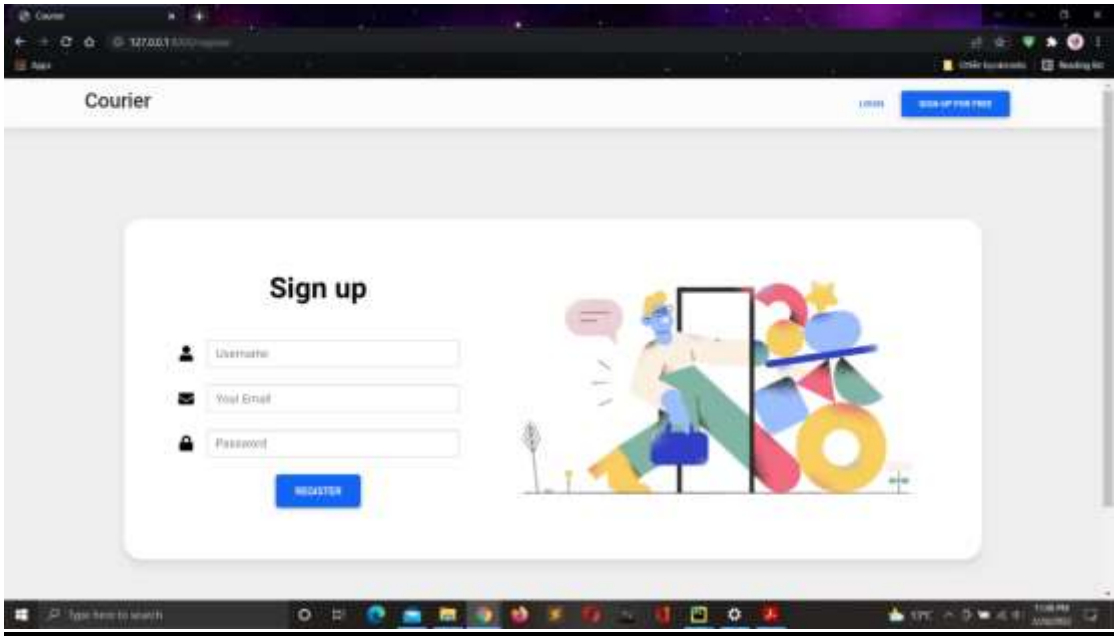


Figure 7: Registration Page

Login Screen

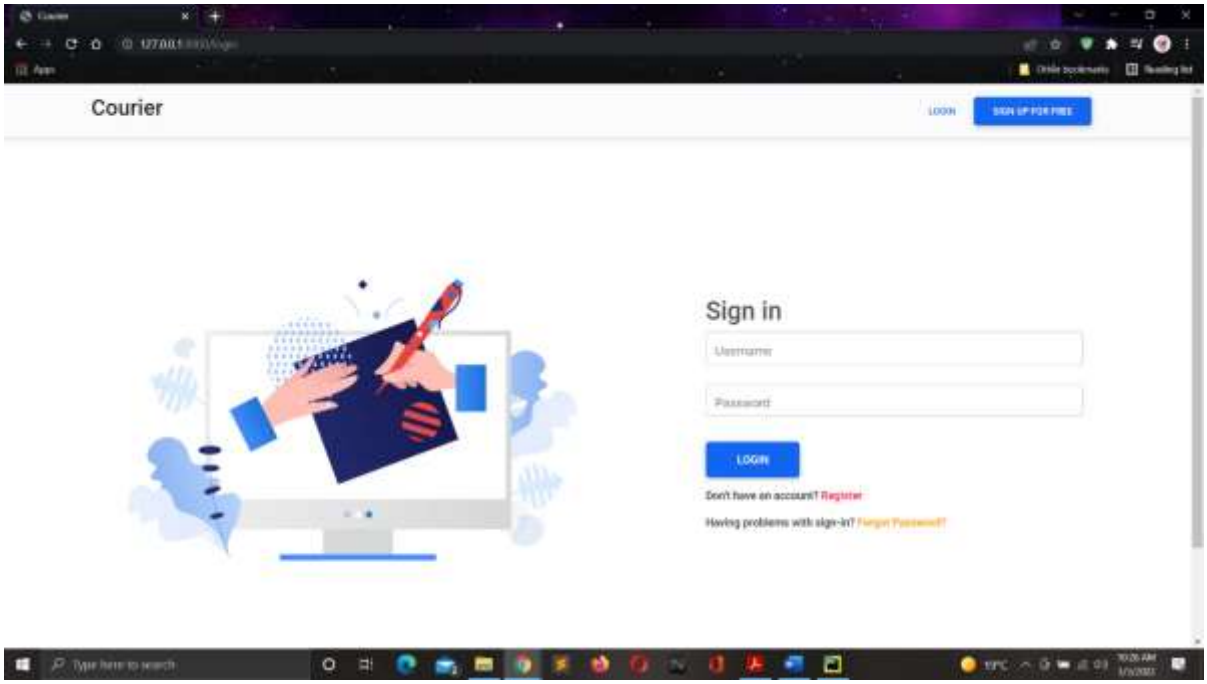


Figure 8: Login Page

Courier List Screen

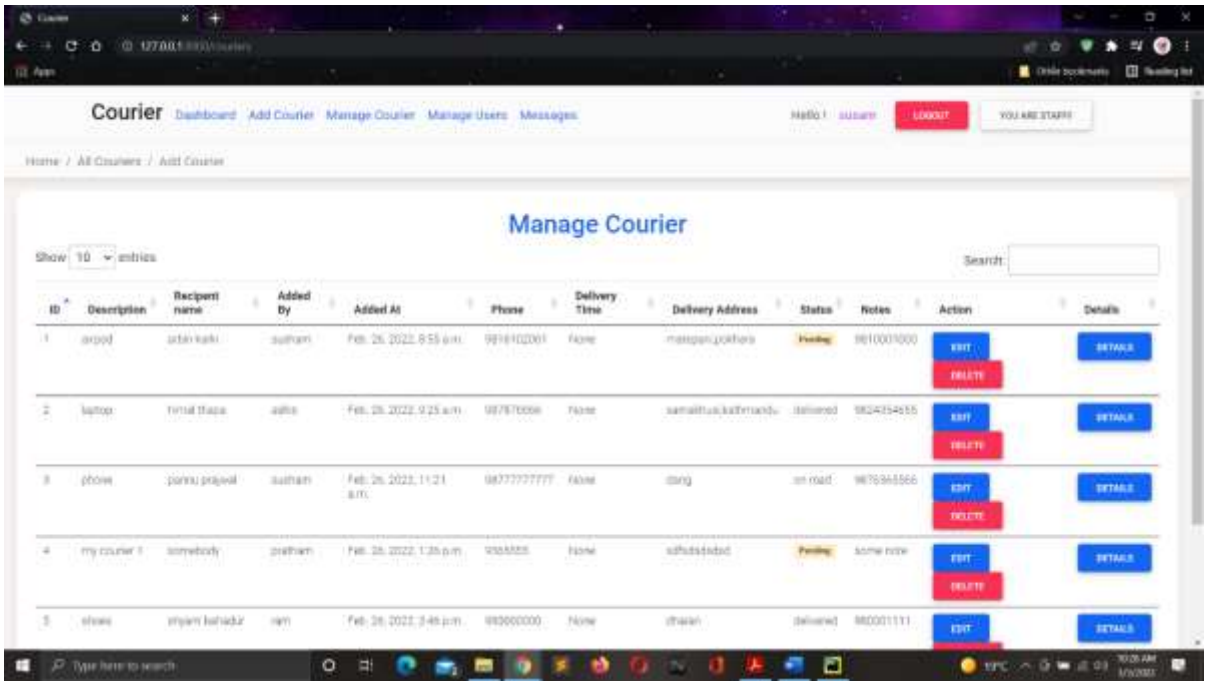


Figure 9: Courier List

Source Code:

DELIVERY MODEL:

```
from django.contrib import admin

# Register your models here.
from .models import Courier


class CourierAdmin(admin.ModelAdmin):
    list_filter = ["status", "created_at"]
    list_display = ["pk", "user", "status", "recipent_name",
"created_at"]
    search_fields = ["user__username"]
    list_per_page = 10


admin.site.register(Courier, CourierAdmin)



from django.db import models
from django.shortcuts import reverse
from phonenumber_field.modelfields import PhoneNumberField
# Create your models here.
#Models for courier management system


class Courier(models.Model):
    user = models.ForeignKey('auth.User', on_delete=models.CASCADE)
    description = models.TextField()
    recipent_name = models.CharField(max_length=100)
    phone = models.CharField(max_length=100, )
    delivery_time = models.DateTimeField(blank=True, null=True)
    delivery_address = models.CharField(max_length=100)
    notes = models.TextField()
    # implement choices
    status = models.CharField(max_length=100, default="Pending",
blank=False, null=False, choices=[])
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return str(self.description)

    def get_absolute_url(self):
        return reverse('Delivery:couriers')



from django.shortcuts import render, redirect
```

```
from .models import Courier
from .forms import CourierForm
from django.contrib.auth.mixins import LoginRequiredMixin  ,
UserPassesTestMixin
from django.views.generic.edit import UpdateView , CreateView
from django.views.generic import ListView, DetailView
from django.utils import timezone


# Create your views here.
#listview for courier
class CourierListView(LoginRequiredMixin , ListView):
    model = Courier
    template_name = 'Delivery/couriers.html'
    context_object_name = 'couriers'
    ordering = ['-created_at']
    def get_queryset(self):
        return Courier.objects.all()

#detailview for courier
class CourierDetailView(LoginRequiredMixin , DetailView):
    model = Courier
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['now'] = timezone.now()
        return context


class  CourierView ( LoginRequiredMixin, UserPassesTestMixin,
CreateView):

    def test_func(self):
        login_url = '/login/'
        return(self.request.user)


    def get(self,request):

        couriers = Courier.objects.filter(user=self.request.user)
        context = {
            'couriers': couriers,
        }
        return render(request, 'Delivery/courier.html', context)

    def post(self, request):
        courier_form = CourierForm(request.POST , request.FILES)
        context = {
            'courier_form' : CourierForm,
        }

        if courier_form.is_valid():
            courier_form.save()
            return redirect('Delivery:courier')
        else:
```

```
            print(courier_form.errors)
            return render(request, 'Delivery/couriers.html', context)


        return redirect('Delivery:courier')

def delete_courier(request , id):
    Courier.objects.get(id=id).delete()
    render(request , 'Delivery/courier.html' , {'response' : "Courier
deleted successfully"})
    return redirect('Delivery:couriers')

class UpdateCourier( UpdateView):

    def test_func(self):
        login_url  = 'login/'
        return(self.request.user)

    model = Courier

    fields = ('__all__')

from django.urls import path
from django.contrib import admin
from . import views
urlpatterns = [
    path('courier', views.CourierView.as_view(), name='courier'),
    path('couriers', views.CourierListView.as_view(), name='couriers'),
    path('update-courier/<int:pk>', views.UpdateCourier.as_view(),
name='update-courier'),
    path('delete-courier/<int:id>', views.delete_courier, name='delete-
courier'),
    path('courier-detail/<int:pk>', views.CourierDetailView.as_view(),
name='courier-detail'),

]

from django import forms

from .models import Courier

class CourierForm(forms.ModelForm):
    class Meta:
        model = Courier
        fields = '__all__'
        exclude = ['created_at', 'updated_at']

from django.apps import AppConfig


class DeliveryConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'Delivery'
```

USER MODEL:

```
from django.contrib.auth.decorators import login_required
from django.shortcuts import render, redirect
from .forms import UserForm
from django.contrib.auth.models import User
from django.contrib.auth import authenticate , login , logout
from django.db.models import Q
from Delivery.models import Courier


# Create your views here.
def index(request):
    if request.user.is_authenticated:
        recent_couriers =
Courier.objects.filter(user=request.user).order_by('-id')[:5]
        courier_count = Courier.objects.all().count()
        users_count = User.objects.all().count()
        pending_couriers =
Courier.objects.filter(status='Pending').count()
        delivered_couriers =
Courier.objects.filter(status='Delivered').count()

        context = {
            'courier_count': courier_count,
            'users_count': users_count,
            'pending_couriers': pending_couriers,
            'delivered_couriers': delivered_couriers,
            'recent_couriers': recent_couriers,
        }
        return render(request, 'User/index.html', context)

    else:
        return redirect('User:login')


# search courier by any field
def search_courier(request, search_query):
    if request.user.is_authenticated:

        search_query = str(search_query)
        # filter data by search query and search in all fields
        couriers = Courier.objects.filter(
        Q(status__icontains=search_query) |
        Q(user__username__icontains=search_query) |
        Q(delivery_time__icontains=search_query) |
        Q(delivery_address__icontains=search_query) |
        Q(notes__icontains=search_query) |
        Q(description__icontains=search_query)  |
        Q(recipent_name__icontains=search_query) |
        Q(phone__icontains=search_query) |
        Q(notes__icontains=search_query)
```

```
        )
        context = {
            'couriers': couriers,
        }
        return render(request, 'Delivery/search_courier.html', context)
    else:
        return redirect('User:login')


def register_user(request):
    if request.method == 'POST':
        form = UserForm(request.POST)
        if form.is_valid():
            form.save()
            user =
User.objects.get(username=request.POST.get('username'))
            login(request, user)

            #add user log
            return redirect('/')

        else:
            print(form.errors)
            context = {
                'error' : form.errors,
            }
            return render(request, 'User/register.html', context)
    else:
        return render(request, 'User/register.html')


def login_user(request):
    if request.user.is_authenticated:
        if request.user.is_superuser:
            return redirect('User:index')
        return redirect('/customer')
    else:
        if request.method == 'POST':
            username = request.POST.get('username')
            password = request.POST.get('password')

            # if username or password is epmty return error for login
page

            user = authenticate(request, username=username,
password=password)
            if user is not None:
                login(request, user)
                #add user log
                return redirect('/user/login.html')
            else:
                # show unauthenticated message in login page
```

```
                pass

        else:
            return render(request, 'User/login.html')

@login_required
def logout_user(request):

    #add user log
    logout(request)
    return redirect('User:login')

#users list
def users_list(request):
    if request.user.is_authenticated and request.user.is_superuser:
        users = User.objects.all()
        #get courier count of every users and append to users
        for user in users:
            user.courier_count =
Courier.objects.filter(user=user).count()

        context = {
            'users': users,
        }
        return render(request, 'User/users.html', context)
    else:
        return redirect('User:login')

#toggle user active status
def toggle_user_active(request, id):
    user = User.objects.get(id=id)
    if user.is_active:
        user.is_active = False
    else:
        user.is_active = True
    user.save()
    return redirect('User:users')

#toggle user admin status
def toggle_user_admin(request, id):
    user = User.objects.get(id=id)
    if user.is_superuser:
        user.is_superuser = False
    else:
        user.is_superuser = True
    user.save()
    return redirect('User:users')

#delete user
def delete_user(request, id):
    user = User.objects.get(id=id)
    user.delete()
    return redirect('User:users')
```

```
from unicodedata import name
from django.urls import path
from . import views

urlpatterns = [
   path('', views.index, name='index'),
    path('login', views.login_user, name='login'),
    path('register', views.register_user, name='register'),
    path('logout', views.logout_user, name="logout"),
    path('users', views.users_list, name='users'),
    path('toggle-user-active/<int:id>', views.toggle_user_active,
name='toggle_user_active'),
    path('toggle-user-admin/<int:id>', views.toggle_user_admin,
name='toggle_user_admin'),

    path('search/<str:search_query>', views.search_courier,
name='search'),

]


from django import forms

from django.contrib.auth.models import User

class UserForm(forms.ModelForm):
    class Meta:
        model = User
        fields = ('id', 'username', 'email', 'password')
        extra_kwargs = {'password': {'write_only': True}}
    def save(self):
        user = User.objects.create_user(
            self.cleaned_data['username'],
            self.cleaned_data['email'],
            self.cleaned_data['password']
        )
        return user


from django.apps import AppConfig


class UserConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'User'


CUSTOMER MODEL:


from django.shortcuts import render, redirect
from Delivery.models import Courier
from Delivery.forms import CourierForm
```

```
from django.contrib.auth.mixins import LoginRequiredMixin  ,
UserPassesTestMixin
from django.views.generic.edit import UpdateView , CreateView
from django.views.generic import ListView, DetailView
from django.utils import timezone
# Create your views here.

def index(request):
    recent_couriers = Courier.objects.order_by('-
created_at').filter(user=request.user)[:3]
    courier_count = Courier.objects.filter(user=request.user).count()
    pending_couriers =
Courier.objects.filter(user=request.user).filter(status='Pending').coun
t()
    delivered_couriers =
Courier.objects.filter(user=request.user).filter(status='Delivered').co
unt()

    print(recent_couriers)


    context = {
        'recent_couriers': recent_couriers,
        'courier_count': courier_count,
        'pending_couriers': pending_couriers,
        'delivered_couriers': delivered_couriers,

    }
    return render(request, 'Customer/index.html', context)

#listview for courier
class CourierListView(LoginRequiredMixin , ListView):
    model = Courier
    template_name = 'Delivery/couriers.html'
    context_object_name = 'couriers'
    ordering = ['-created_at']
    def get_queryset(self):
        return Courier.objects.filter(user=self.request.user)



class  CourierView ( LoginRequiredMixin, UserPassesTestMixin,
CreateView):

    def test_func(self):
        login_url = '/login/'
        return(self.request.user)


    def get(self,request):

        couriers = Courier.objects.filter(user=self.request.user)
        context = {
            'couriers': couriers,
```

```
        }
        return render(request, 'Customer/courier.html', context)

    def post(self, request):
        courier_form = CourierForm(request.POST , request.FILES)
        context = {
            'courier_form' : CourierForm,
        }

        if courier_form.is_valid():
            courier_form.save()
            return redirect('Customer:courier')
        else:
            print(courier_form.errors)
            return render(request, 'Customer/couriers.html', context)


        return redirect('Customer:courier')



from django.urls import path
from . import views
urlpatterns = [
    path('customer', views.index, name='customer_index'),
    path('customer-couriers', views.CourierListView.as_view(),
name='courier_list_customer'),
    path('customer-courier', views.CourierView.as_view(),
name='courier_customer'),
]


from django.apps import AppConfig


class CustomerConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'Customer'
```