CHINMAYA ARTS AND SCIENCE COLLEGE FOR WOMEN

GOVINDAGIRI, CHALA, KANNUR

(Affiliated to Kannur University and approved by Govt. of Kerala)

SIXTH SEMESTER BCA
2024-25
PROJECT REPORT

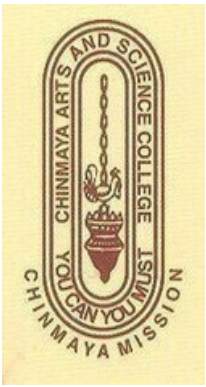
SCRAPPY

Prepared and Presented by

HALEEMA M
SHADHA SOOPPY
ANUSREE BAIJU

CHINMAYA ARTS AND SCIENCE COLLEGE FOR WOMEN

GOVINDAGIRI, CHALA, KANNUR



CERTIFICATE

This is to certify that the project entitled 'SCRAPPY' is a bonafide report of the project work done by Shadha Soopy, Reg-No CW22BCAR32, carried out under the supervision of Ms Jasna B, towards partial fulfillment of the award of BCA degree at Kannur University.

Guide	Head of the Department
Ms Jasna B	Ms Anitha Haridas
External Examiners:	Principal

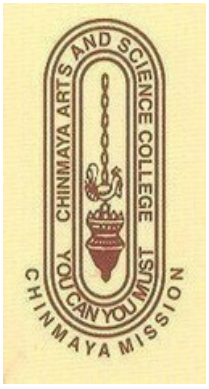
1.

2.

Submitted for practical examination held on .....

CHINMAYA ARTS AND SCIENCE COLLEGE FOR WOMEN

GOVINDAGIRI, CHALA, KANNUR



DECLARATION

I, Shadha Soopy, Reg-No CW22BCAR32 VI Semester BCA student of Chinmaya Arts and Science College for Women, do hereby declare that the project report entitled 'SCRAPPY' is the original work carried out by me under the supervision of Ms Jasna B towards the partial fulfillment for the requirement of BCA Degree at Kannur University.

Signature and Name of the student:

Date of Submission:.....
Kannur:

ABSTRACT

The central objective of our application is to provide a convenient, efficient, and organized platform for users to collect, manage, and recycle scrap materials, thereby promoting sustainable waste disposal practices and reducing environmental impact. “Scrappy” is an innovative application designed to streamline scrap collection and recycling processes. This aims to bridge the gap between scrap generators and collectors, facilitating a seamless and efficient scrap management experience. By leveraging GPS tracking, real- time scheduling, and in-app payment systems, “scrappy” enhances user convenience, reduces logistical complexities, and promotes environmentally responsible waste disposal practices. By simplifying the scrap collection process, the application aims to increase recycling rates, reduce waste send to landfills, and promote sustainable waste management. Furthermore, “Scrappy” seeks to educate users about the importance of recycling, proper waste segregation, and the environmental benefits of responsible scrap disposal. The application’s intuitive interface enables users to easily schedule collections, track payments, and access recycling information, making it an indispensable tool for individuals, households, and businesses seeking to manage their scrap responsibly. Moreover, Scrappy’s robust backend infrastructure ensures a reliable, secure, and scalable platform for all users

ACKNOWLEDGMENT

I humbly acknowledge the divine guidance that led me to complete the project. The support and timely direction received greatly contributed to the project's success. Special thanks to Dr. Seema M Thayil, our principal, for granting us permission and facilitating the project. Gratitude to Ms. Anitha Haridas, HOD of the Computer Application Department, for her assistance. We extend our thanks to Mrs. Jasna B for her invaluable guidance and continuous cooperation. Our guide's advice played a crucial role in the successful completion of the project. We also thank our guides at Riss Technologies, Kannur for their guidance and valuable suggestions in bringing out this project successfully. We also appreciate the encouragement and assistance from friends, family, and Mrs. Shamna P V, the Lab assistant. Thanks to colleagues for their cooperation and support.

CONTENTS

1. INTRODUCTION

SCRAPPY

Scrappy serves as a vital platform connecting individuals, households, and businesses with scrap collectors and recyclers, facilitating the efficient disposal of waste materials. Through the app, users can schedule collections, identify scrap types and prices, and track payments, making the process convenient and transparent. Scrappy plays as a game- changer in the waste management industry, offering a one-stop solution for individuals, households, and businesses to dispose scrap materials. With features such as scheduling collections, tracking payments, and accessing recycling information, the app provides user- friendly experience, encouraging users to adopt environmentally conscious habits and contribute to a cleaner, greener future. Scrappy is interconnected with a network of stakeholders, including scrap collectors and recyclers, recycling facilities, and recycling factories. Through these connections, the app facilitates the efficient collection, processing, and recycling of scrap materials, promoting environmentally responsible waste disposal practices. We provide the best value for your scrap from our network of recyclers, doorstep pickup and delivery, trained and verified pickup staff. We ensure responsible recycling of your scrap items.

OBJECTIVE

The primary objective of scrappy is to streamline scrap collection, promote sustainable waste management, and connect users with reliable scrap collection. By achieving these goals, the application aims to increase recycling rates, reduce environmental impacts, and support local communities in their efforts to manage waste responsibly. To incentivize users and promote sustainable waste management practices, scrappy offers a range of exciting rewards and discounts. Users can earn points for every scrap collection, redeemable for discounts on future collections or exclusive deals. By combining convenience, education, and rewards, the app empowers users to take control of their waste management and contribute to a more sustainable future

2.SYSTEM STUDY

2.1 SCOPE

The scope of this application is to streamline the process of scrap collection, management, recycling, and selling new recycled products. The application will enable users to register and list their scrap materials, request collection services, schedule appointments, and buy new products. It will also facilitate payment processing, generate reports on collection activities, provides analytics on scrap material management, and integrate with google maps for location-based services and social media platforms for user engagement. The application will prioritize data security and compliance with relevant regulations, ensuring a seamless and reliable experience for users.

2.2 PRELIMINARYSTUDY

As part of the studies for our application, we went through various websites and applications.Some of these are given below:

- Chat GPT
- Google
- Scrap Uncle (both website and application are available)
- ScrapZoo (both website and application are available)

These websites and applications helped us to get a better view of what our application is going to do.

2.3 EXISTING SYSTEM

- Scrap Uncle : An application that offers a digital platform for scrap material collection and recycling.

Some of the issues facing by this app are:

- Limited payment options
- Inability to handle high volumes
- Lack of transparency in pricing

2.4 PROPOSEDSYSTEM

• Flexible payment options, provides credit points for users and recycle factories

• Accepts all types of waste materials, regardless of volume

• Offers best prices

2.5 FEASIBLITYSTUDY

Feasibility studies are conducted for the development of a successful scrap collection application. By assessing the economical, technical, and operational viability of the application, we can identify potential challenges and opportunities, and determine the best approach to development. The study will evaluate the demand for scrap collection services, assess the competition, and determine the most effective revenue models. Moreover, it will enable us to develop a comprehensive plan for the application's development, launch, and maintenance, ensuring that it meets the needs of users and stakeholders, while also ensuring its financial sustainability

ECONOMICAL FEASIBILITY

The system we developed not only met all requirements but also generates revenue and operational costs. By providing a platform for users to schedule scrap collections, the application can earn commission-based fees from recycle factories. The application's digital platform reduces the need for paperwork, and phone calls, resulting in cost savings for scrap collectors and recycle factories.

TECHNICAL FEASIBILITY

This system is technically feasible and can be implemented successfully without the requirement for added software and hardware components. We have implemented measures to safeguard against unexpected damages or losses.

OPERATIONALFEASIBILITY

This systems design allows for straightforward user operation. The application demonstrates operational feasibility through its streamlined processes and efficient logistics. Our service not only excels in quality but also ensures minimal time requirements. The application also facilitates communication between user, scrap collectors, recycle factories, reducing errors, and miscommunications.

3. SOFTWARE REQUIREMENT SPECIFICATION

HARDWARE SPECIFICATION

Choosing the right hardware is crucial for the proper functioning of any software. The size and capacity of the hardware must align with the specific requirements of the software to ensure optimal performance. Mismatched or inadequate hardware can lead to performance issues and system failures, emphasizing the importance of thoughtful hardware selection for a seamless software experience.

Processor : Intel Core i5

Memory:8GB RAM 256GB SSD

SOFTWAREREQUIREMENTS

Choosing software for a system is challenging, involving matching identified requirements with suitable packages. The task is to determine if a specific software aligns with and addresses the system's needs effectively, requiring careful assessment for a seamless integration.

CodingLanguage : Python , Dart

Front End : HTML , CSS

Back End : Python Django , Dart , Flutter

Operating System : Windows

Platform : Android

Tools Used : Pycharm , Android studio

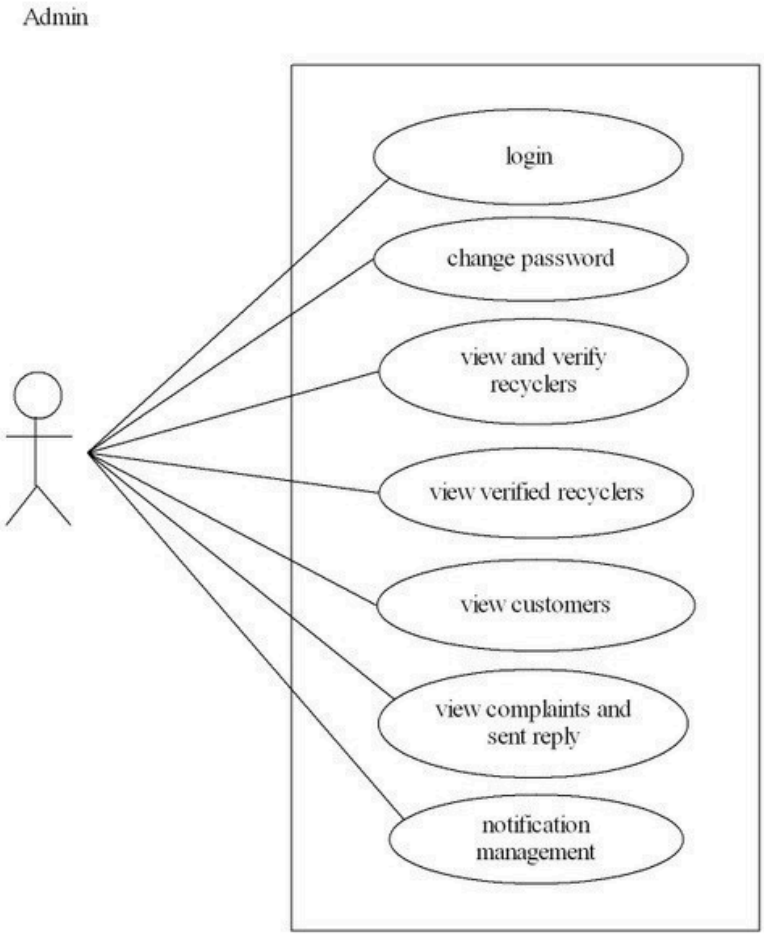
Servers : Wampserver

4. REQUIREMENT SPECIFICATION MODEL

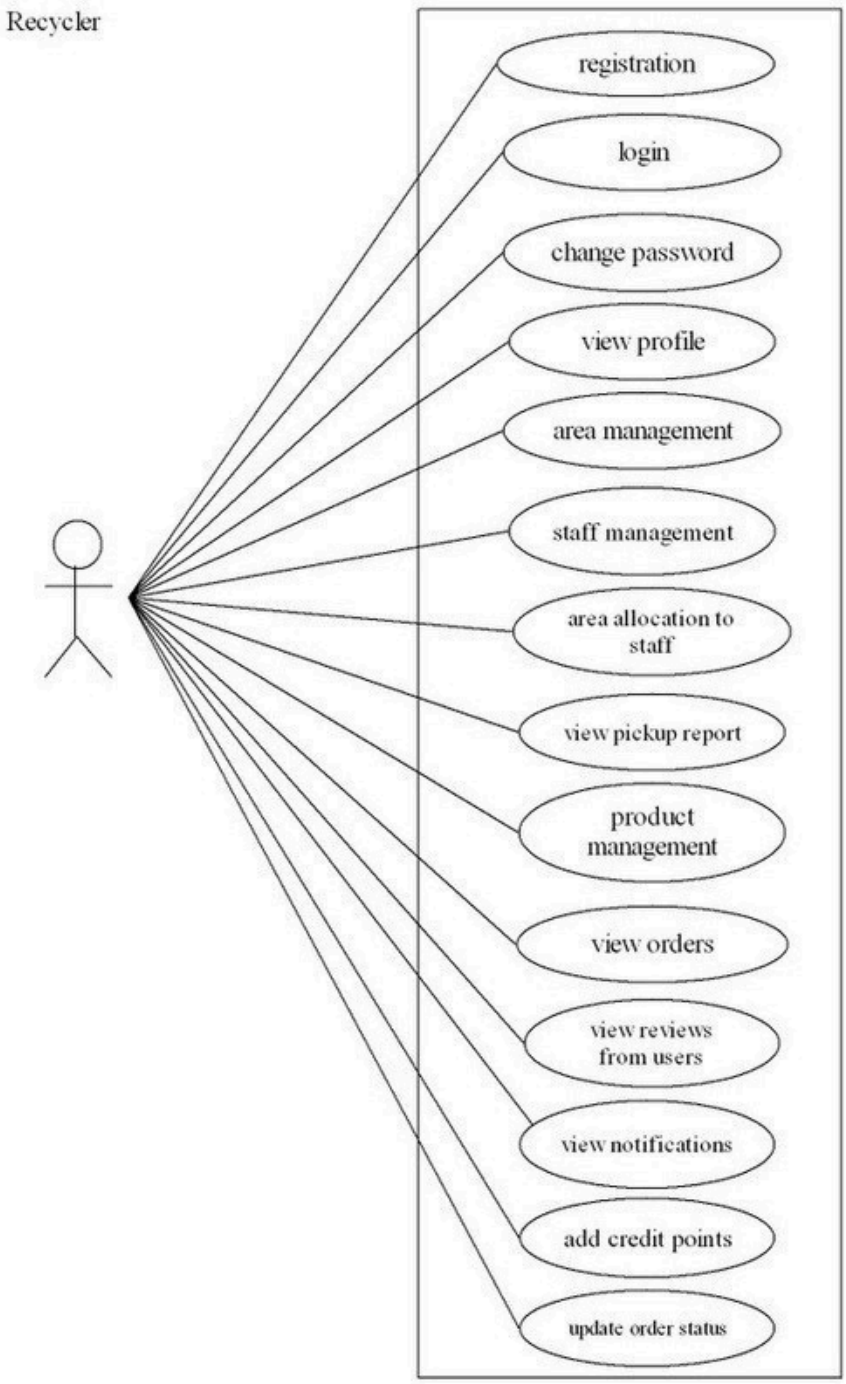
4.1 USECASE DIAGRAM

We have used use case approach in the Unified Modelling Language(UML) to understand the various requirements of each end user. The system is modularized based on users. The main purpose of a use case diagram is to show what system functions are performed by each actor. An actor is a person, organization or an external system that plays a role in one or more interactions of a system .

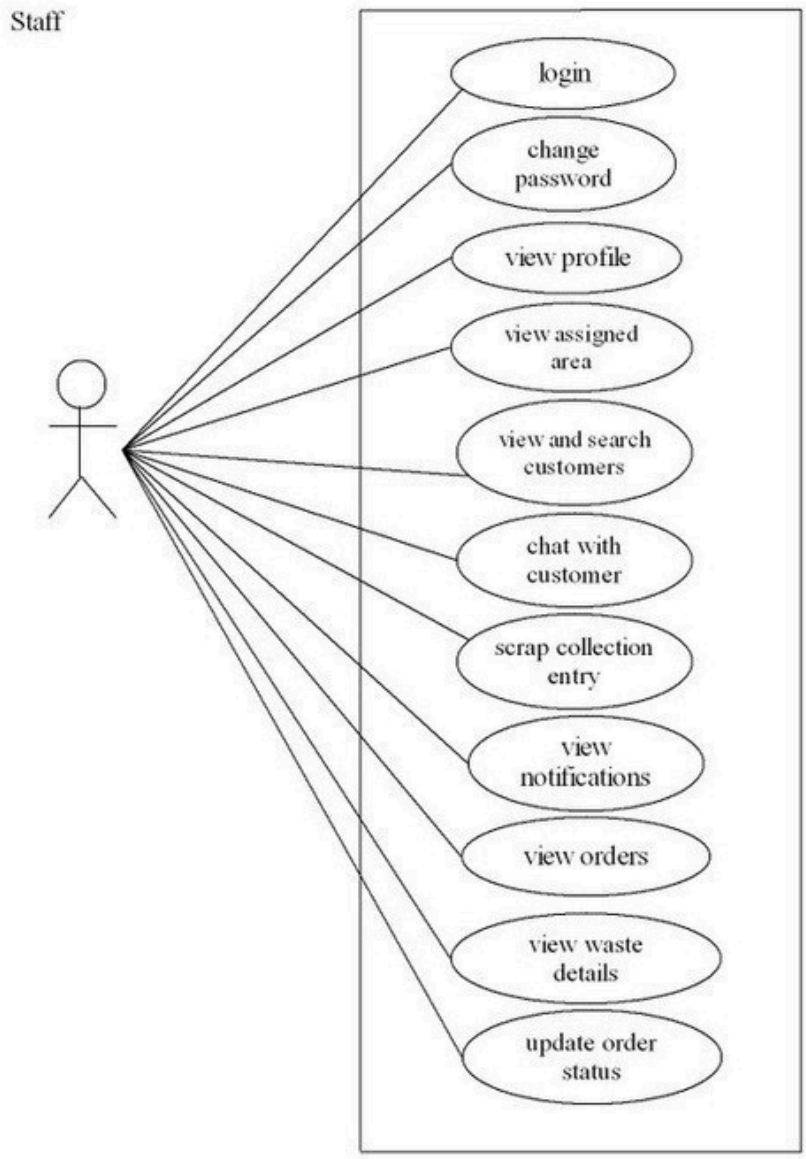
ADMIN



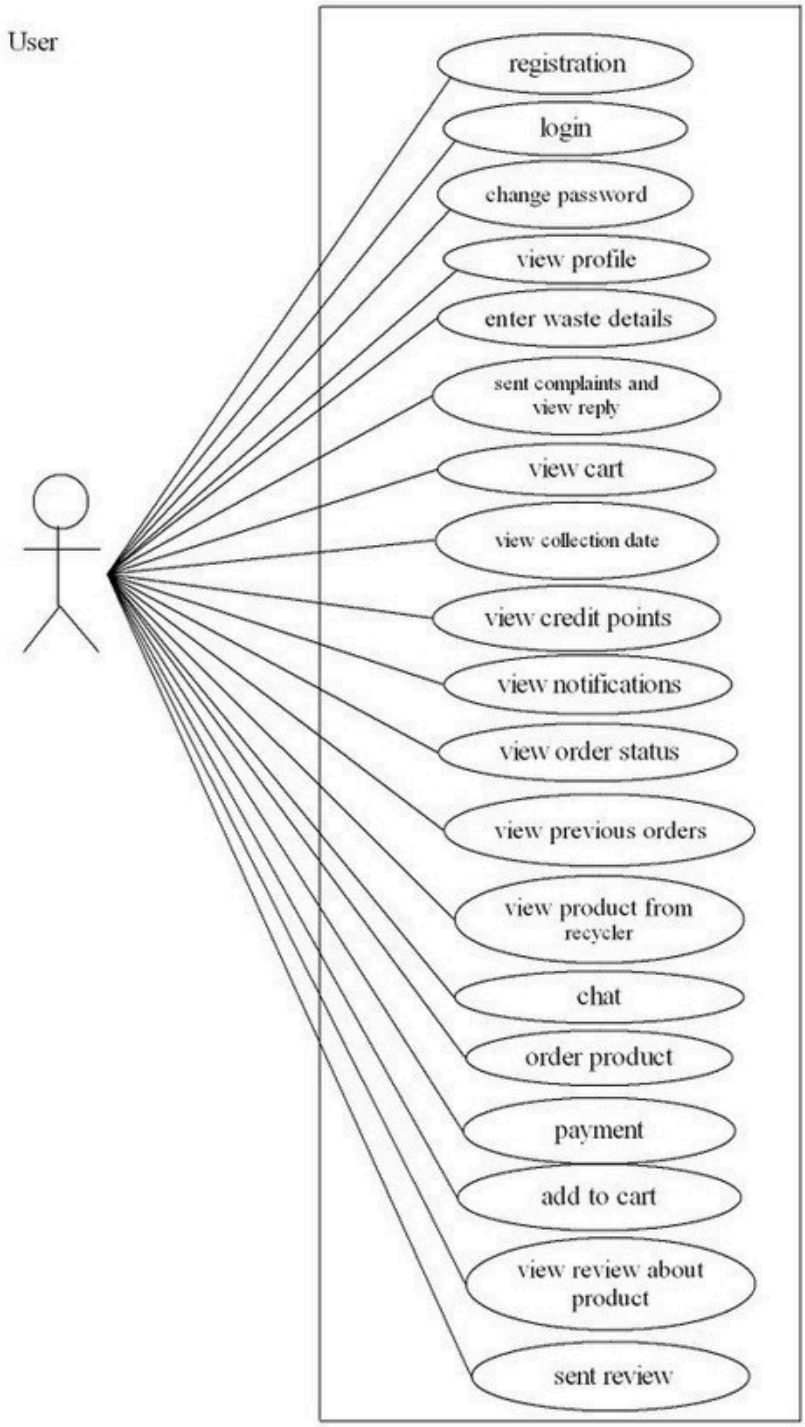
RECYCLER


Recycler
registration
login
change password
view profile
area management
staff management
area allocation to staff
view pickup report
product management
view orders
view reviews from users
view notifications
add credit points
update order status

STAFF


Staff
login
change password
view profile
view assigned area
view and search customers
chat with customer
scrap collection entry
view notifications
view orders
view waste details
update order status

USER

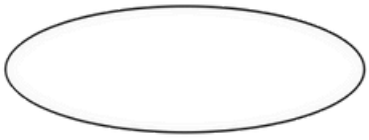

User
registration
login
change password
view profile
enter waste details
sent complaints and view reply
view cart
view collection date
view credit points
view notifications
view order status
view previous orders
view product from recycler
chat
order product
payment
add to cart
view review about product
sent review

4.2 DATAFLOWDIAGRAM

Data flow Diagrams (DFD) is the most commonly used way of documenting the processing of the required system. They are the pictorial way of showing the flow of data into, around and out of the system. They can be understood by the users and are less prone to misinterpretation than textual description. A complete set of DFDs provide a compact top-down representation of the system, which makes it easier for users and analysts to envisage the system as a whole. DFD also known as Bubble Chart has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design phase that functionally decomposes the requirements specifications down to the level of details. It does not show the information about the timing processes or information about whether processes will operate in sequence or in parallel. A DFD shows, what kind of data will be put into and out of the system, where data will come from and go to and where data will be stored. A DFD is often a preliminary way of creating the overview of the system.

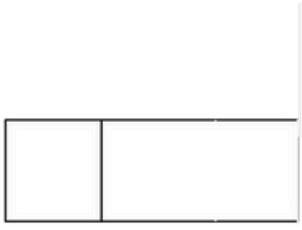
DFD mainly uses the following symbols:

Source/Sink



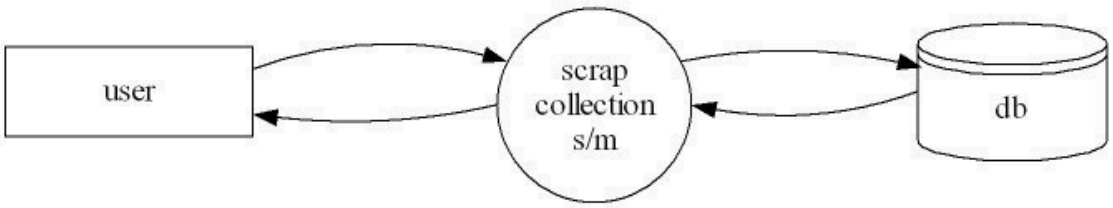
Process

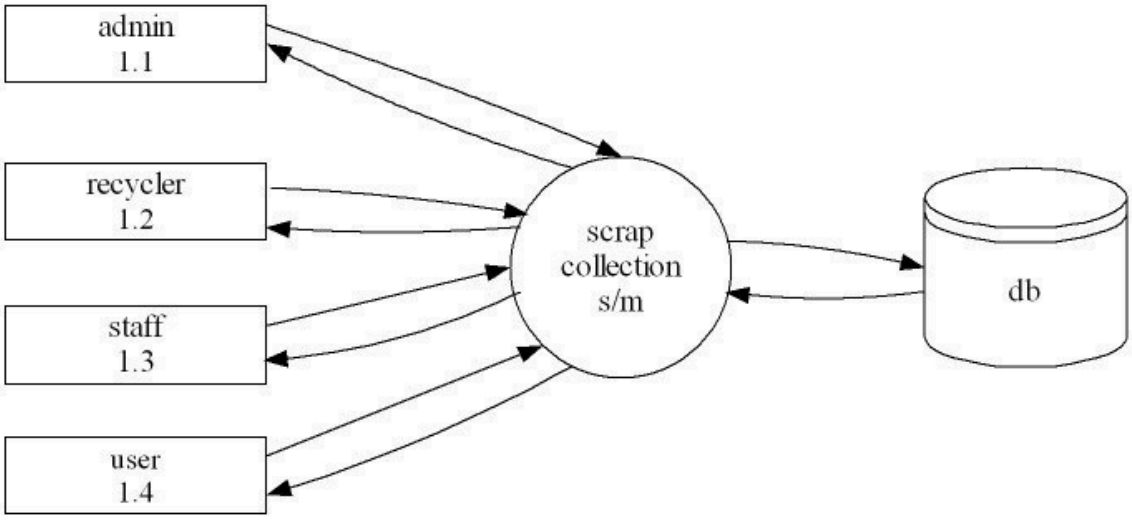
Dataflow



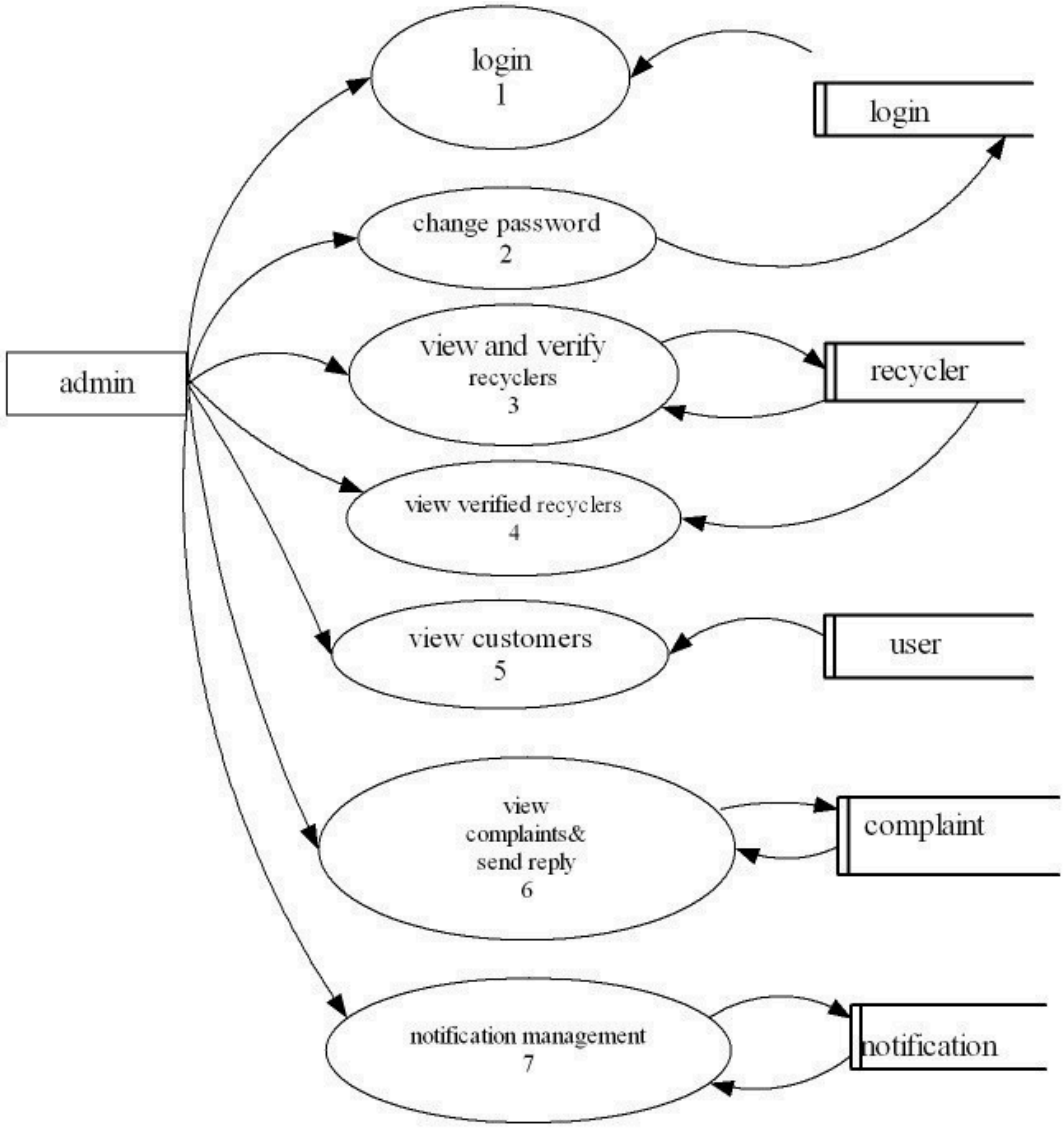
Datastore

LEVEL 0: SCRAPPY

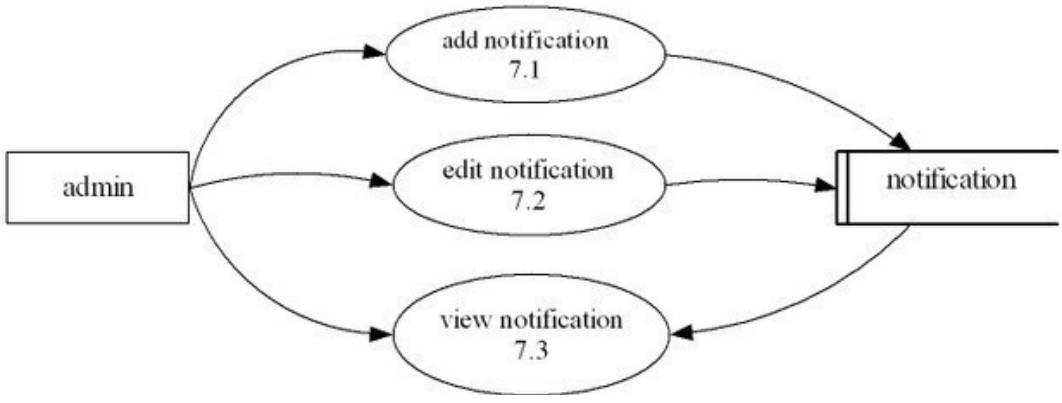




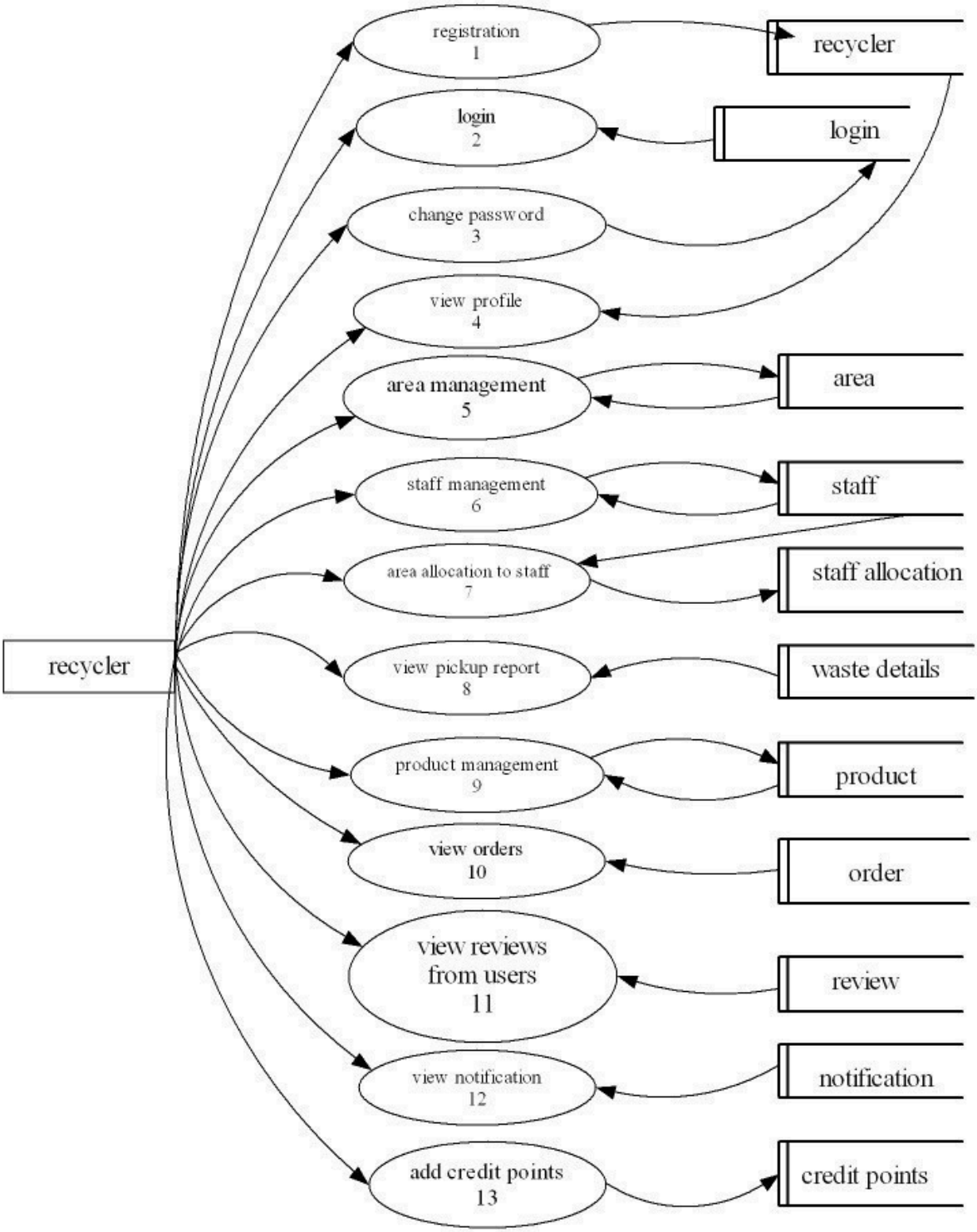
LEVEL 1: ADMIN



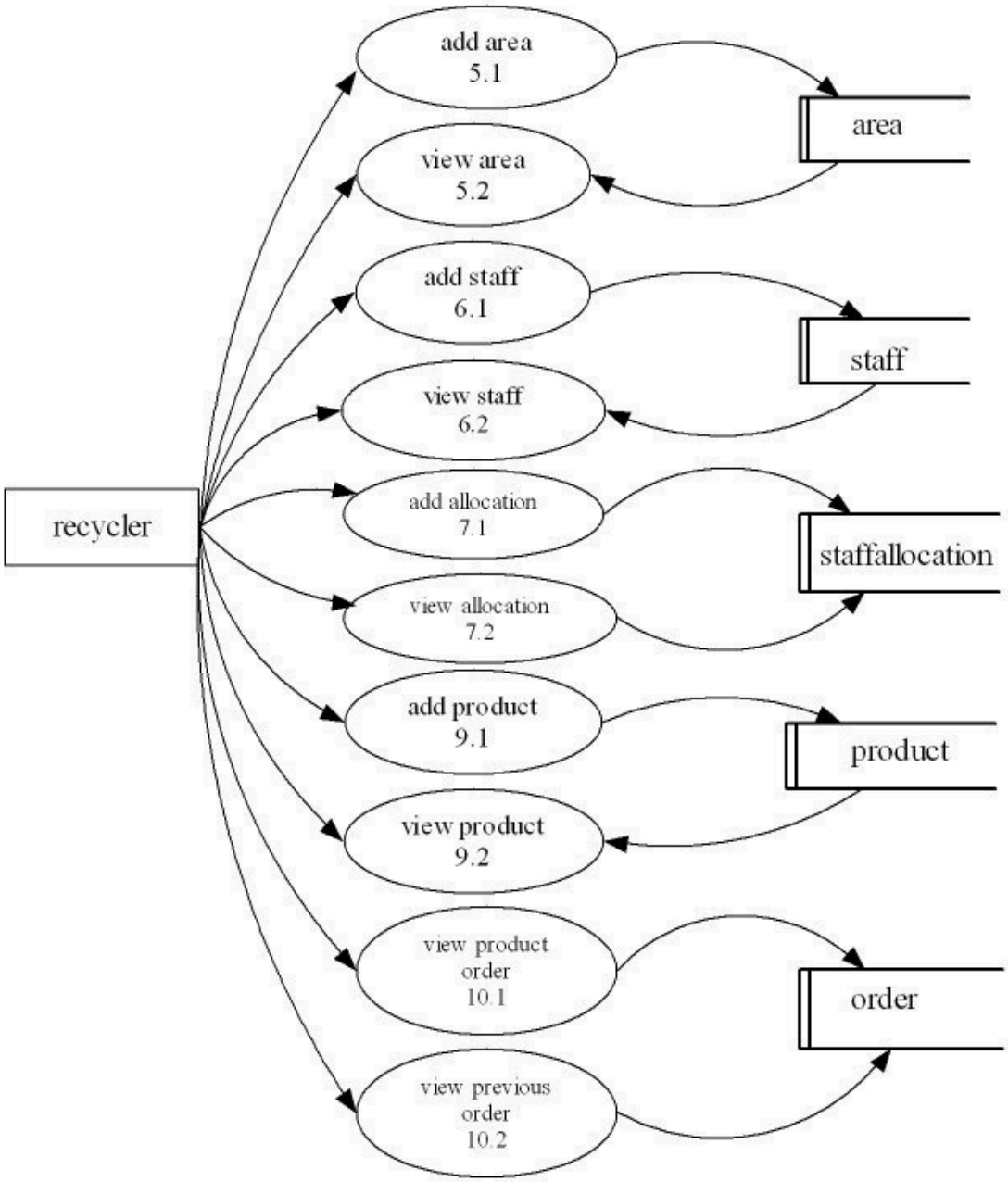
LEVEL 2: ADMIN



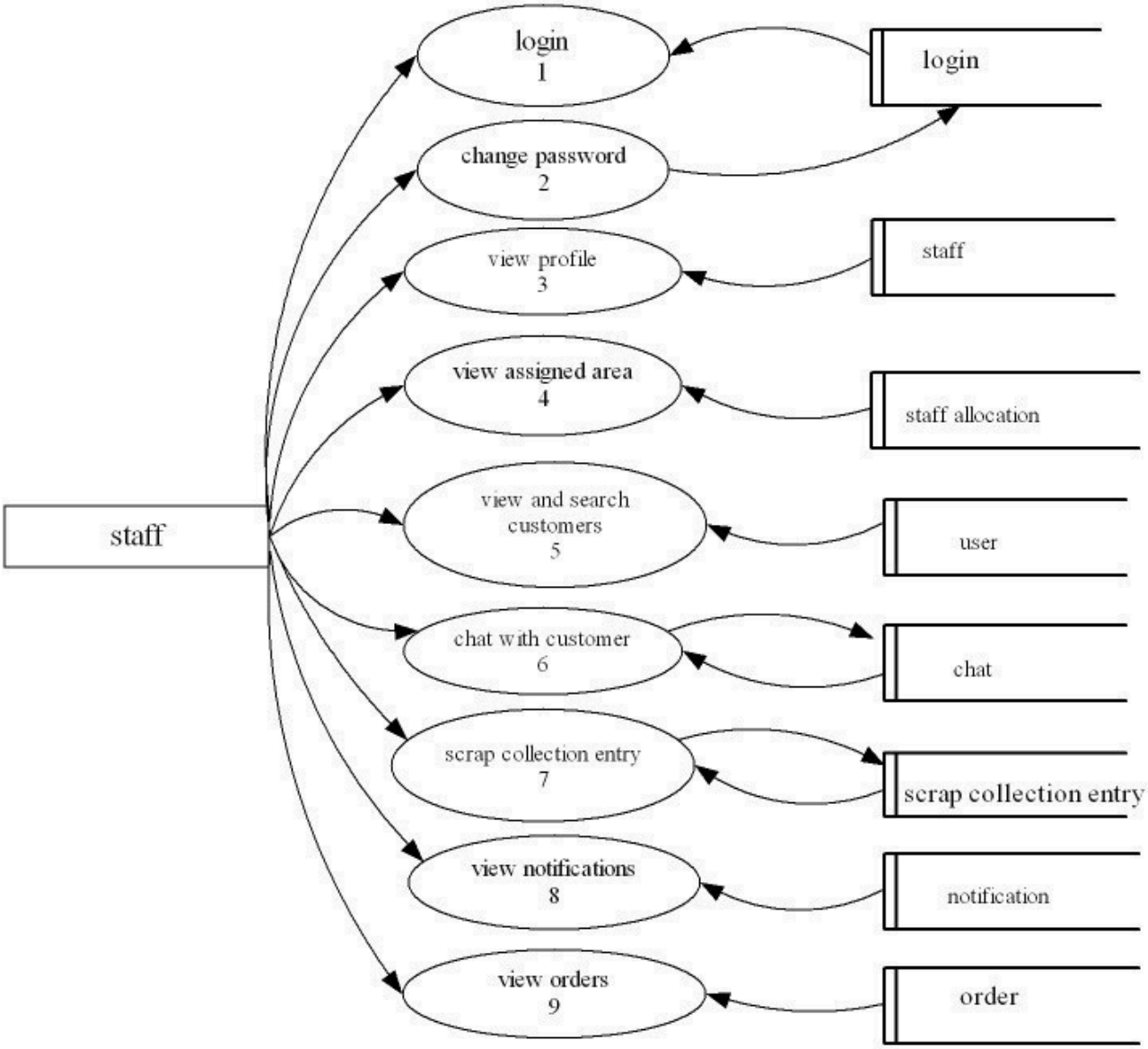
LEVEL 1: RECYCLER



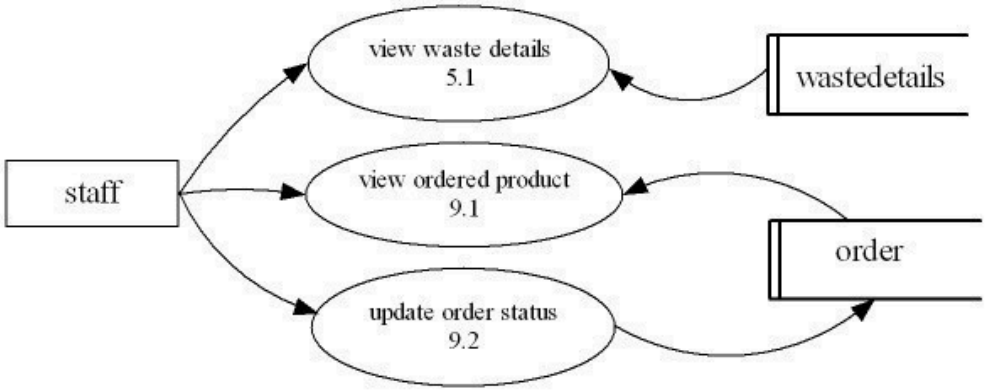
LEVEL 2: RECYCLER



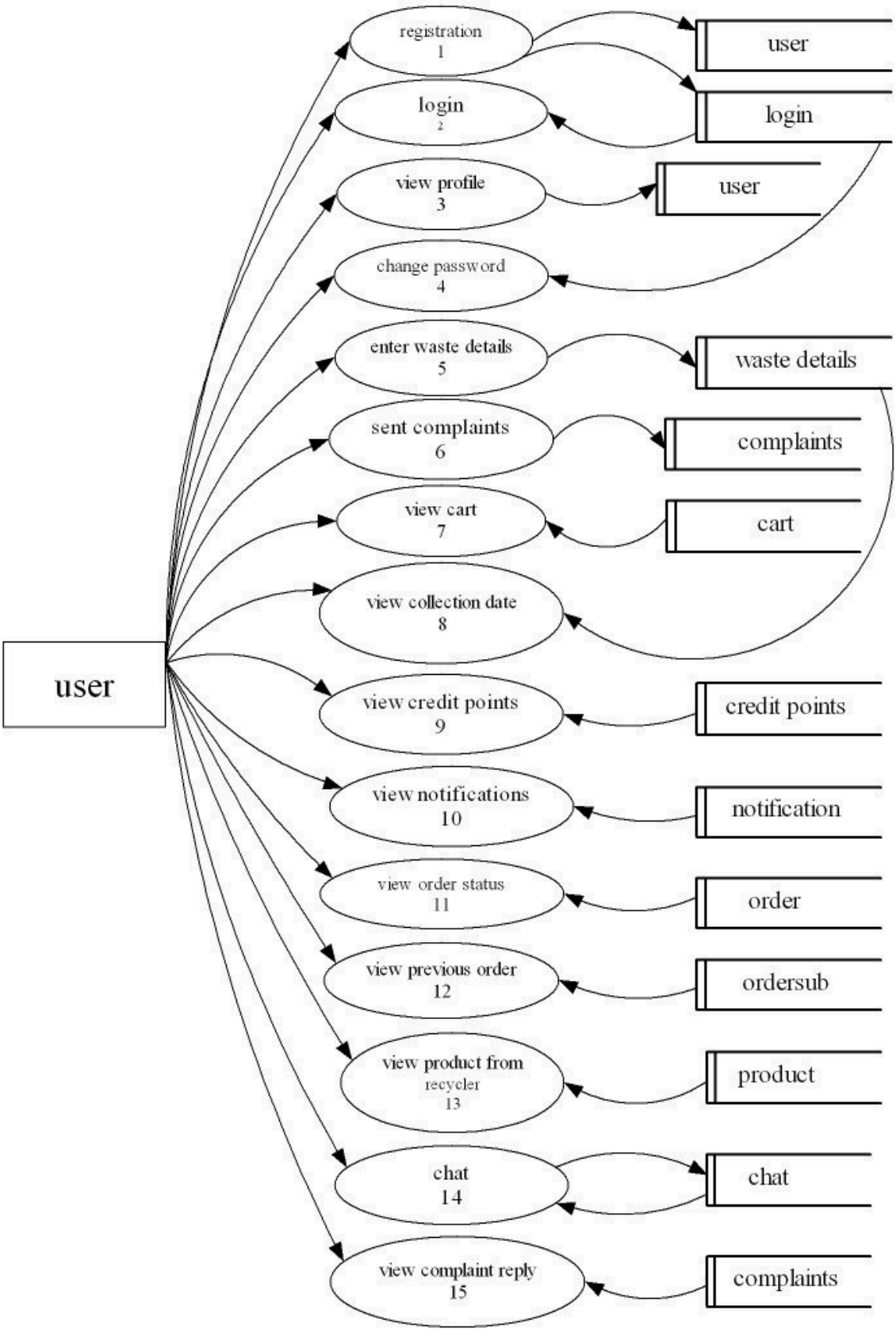
LEVEL 1: STAFF



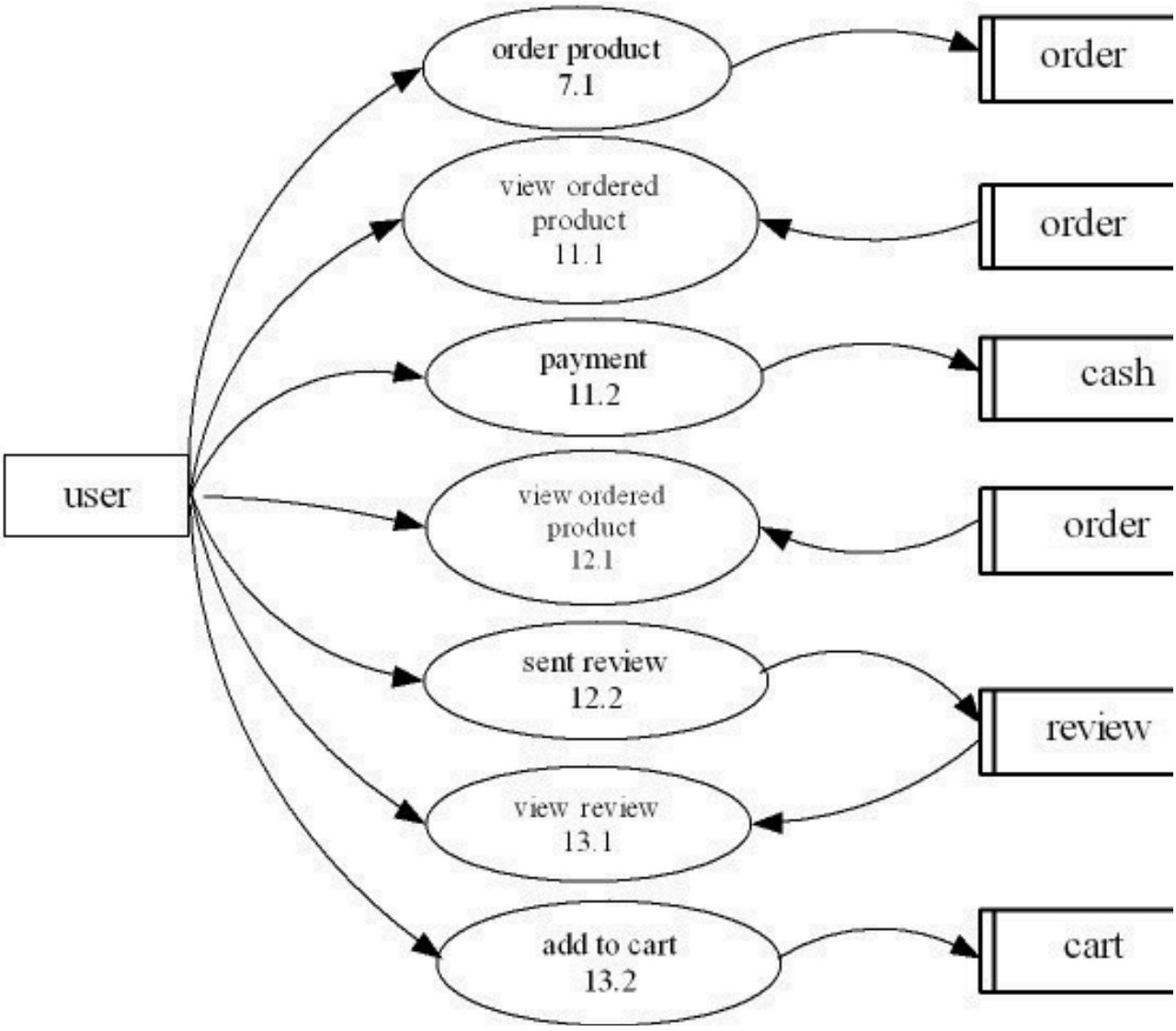
LEVEL 2: STAFF



LEVEL 1: USER


registration
1
login
2
view profile
3
change password
4
enter waste details
5
sent complaints
6
view cart
7
view collection date
8
view credit points
9
view notifications
10
view order status
11
view previous order
12
view product from
recycler
13
chat
14
view complaint reply
15
user
login
user
waste details
complaints
cart
credit points
notification
order
ordersub
product
chat
complaints
user

LEVEL 2: USER


order product
7.1
order
view ordered
product
11.1
order
payment
11.2
cash
user
view ordered
product
12.1
order
sent review
12.2
review
view review
13.1
add to cart
13.2
cart

5. SYSTEM DESIGN

System design, the most creative phase of the development process, serves as the solution to creating the proposed system. It involves crafting technical specifications based on the feasibility study, detailing procedures for implementation. The design specification outlines features, input/output files, and data files, ensuring the system meets requirements for information presentation, accuracy, interaction methods, and overall reliability. Adherence to organizational rules and practices is crucial. Key design objectives encompass practicality, cost efficiency, flexibility, and security, all geared towards fulfilling the specified requirements in the feasibility report.

5.1 MODULES

The proposed system involves 4 modules which are as follows:

1 Admin

2. Recycler

3. Staff

4. User

5.2 HIGH LEVEL DESIGN

ADMIN

Admin has the right to approve and reject the recyclers by checking the uploaded credentials. They can also can view and add different categorizes. Admin can also view the complaints uploaded by the recyclers . They can also view customers.

RECYCLER

Recycler can manage area , staff , and allot areas to staff. They can also add products , edit added products , and delete those products. They can also view pickup reports , product orders , reviews from users , and also add credit points.

STAFF

Staff can view their areas assigned by recycler. They can view and search for customers , view waste details , and add collection date for the waste entered by users. They can view notifications and orders. They can also chat with custom ers.

USER

Users can sell their waste details and earn money or credit points. They can buy products. They can also view products, sent complaints or reviews, view reviews on products, view notifications, order status, and also chat with staff.

5.3 LOWLEVELDESIGN

ADMIN

1. Login : The admin can login to the website by entering username and password. The username and password are stored in the login table.
2. View & verify recycler : The admin can view registered recyclers and verify them.
3. View customers : The admin can view customers in the app.
4. View complaints & send reply : The admin can view complaints and send replies for those complaints.
5. Manage notifications : The admin can add , edit the added notification , or can delete notifications.

RECYCLER

1. Registration : The recycler have to register in the website by giving necessary details.
2. Login : The recycler can login to the website by entering username and password. The username and password are stored in the login table.
3. Area management : The recycler adds area from where they provide services , and also they can remove areas from the added list.
4. Staff management : The recycler can add staffs and remove staffs.
5. Area allocation to staff : The recycler can allocate area for each staff to collect waste materials.
6. Product management : The recycler can add their recycled products to the app , also they can delete the added products.
7. View product order : The recycler can view products ordered by users.
8. View reviews & notifications : The recycler can view product reviews and notifications in the app.
9. Add credit points : The recycles adds credit points in user’s account for each collection.

STAFF

1. Login: The staff can login to the website by entering username and password. The username and password are stored in the login table.
2. View assigned area : The staff can view their assigned areas for collecting waste materials.
3. View & search customers : The staff can view and also search for customers.
4. Chat with customers : The staff can communicate with customers via chat.
5. View notifications : The staff can view notifications.

USER

1. Registration : The user have to register in the website by giving necessary details.
2. Login: The user can login to the website by entering username and password. The username and password are stored in the login table.
3. View credit points : The users can view their credit points credited to their account.
4. View products : The user can view the recycled products added by recyclers and buy them.
5. View reviews & notifications : The users can product reviews given by other users and also the notifications.
6. Add product to cart : The users can add the products to cart and buy them.
7. View orders : The users can view their ordered products , the status of their ordered products , and the previous orders.
8. Send reviews : The users can send reviews for products.
9. Send complaints and view reply : The users can send complaints and also view replies of those complaints.

5.4 ENTITYRELATIONSHIPDIAGRAM

An Entity-Relationship (ER) model utilizes an Entity Relationship Diagram (ER Diagram) to depict the structure of a database. It serves as a design or blueprint that can be implemented as an actual database. The key components of the ER model are entity sets and relationship sets, with an ER diagram illustrating the relationships among entity sets. Entity sets are groups of similar entities, each with attributes. In the context of a Database Management System (DBMS), an entity corresponds to a table or attribute within a table. By showcasing relationships among tables and their attributes, the ER diagram visually represents the complete logical structure of a database. Following are the main components and its symbols in ER Diagrams:-

Rectangles : Represents Entity Set

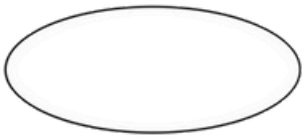
Ellipses : Attributes

Diamonds : Relationship Set

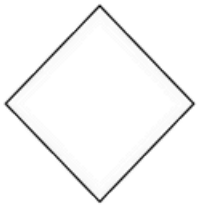
Lines : It links attributes to Entity Set and Entity Set with other Relationship set

Double Ellipses : Multi-valued Attributes

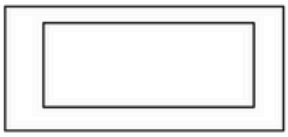
Entity or Strong Entity



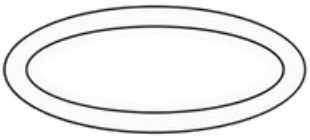
Attributes



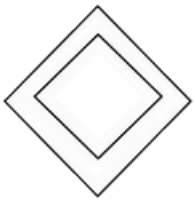
Relationship



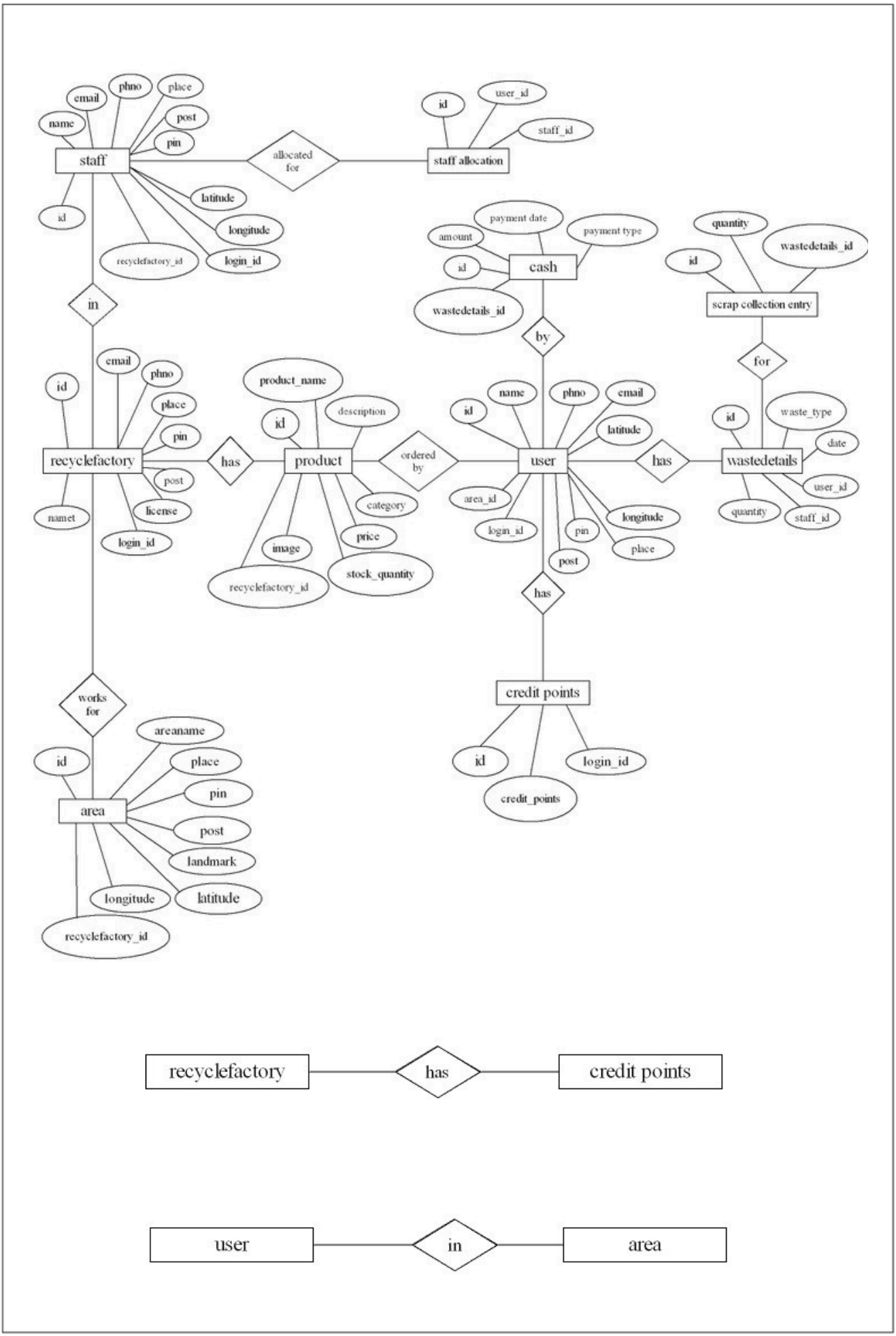
Weak Entity



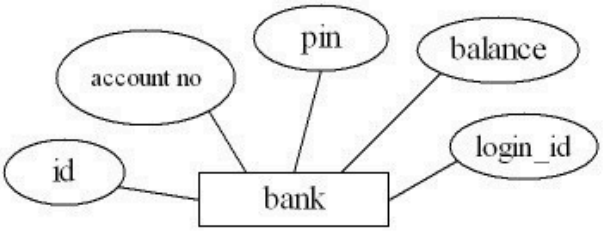
Multi-valued Attributes

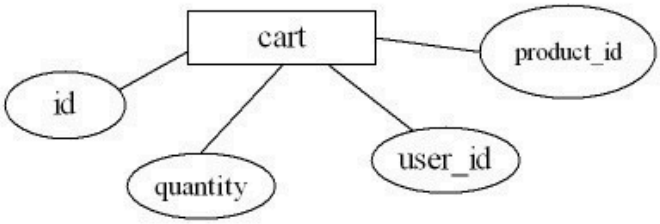


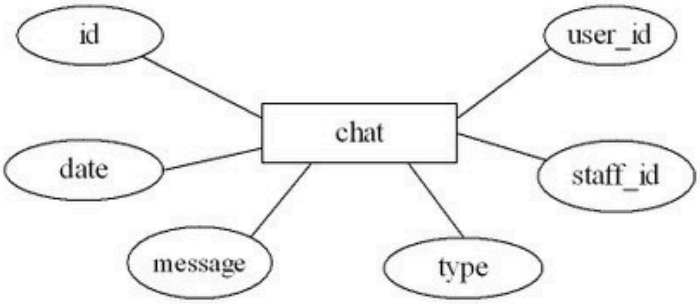
Weak Relationship

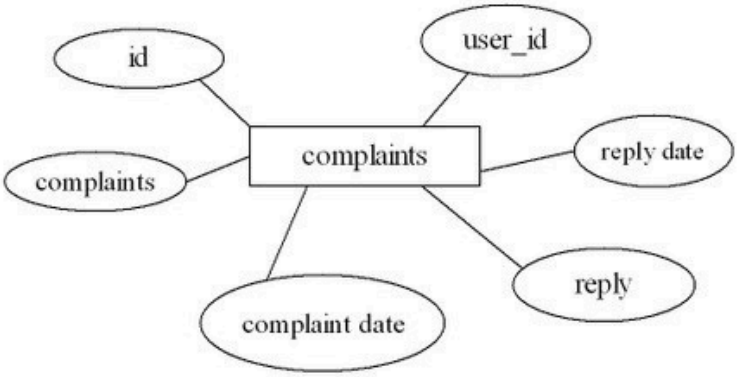

staff
email
phno
place
name
post
pin
id
latitude
longitude
recyclefactory_id
login_id
allocated for
staff allocation
user_id
staff_id
in
recyclefactory
email
phno
id
place
pin
post
namet
license
login_id
has
product
product_name
id
description
category
price
image
stock_quantity
recyclefactory_id
ordered by
user
name
phno
email
id
latitude
area_id
login_id
pin
longitude
place
post
has
credit points
id
login_id
credit_points
payment date
payment type
amount
id
cash
wastedetails_id
by
quantity
wastedetails_id
id
scrap collection entry
for
id
waste_type
date
wastedetails
user_id
quantity
staff_id
has
works for
areaname
id
place
pin
area
post
landmark
longitude
latitude
recyclefactory_id
recyclefactory
has
credit points
user
in
area

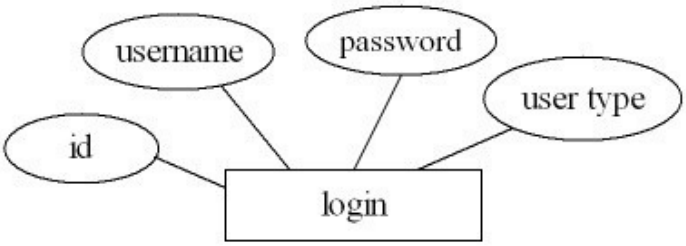
INDEPENDENT ENTITY

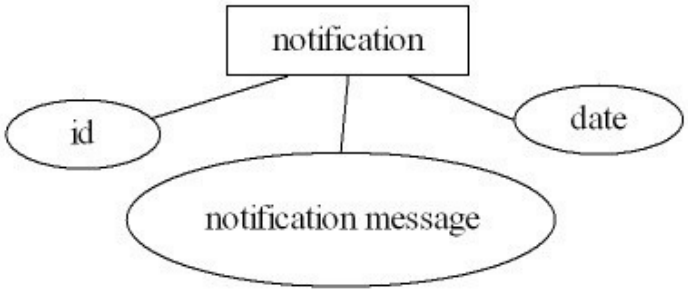


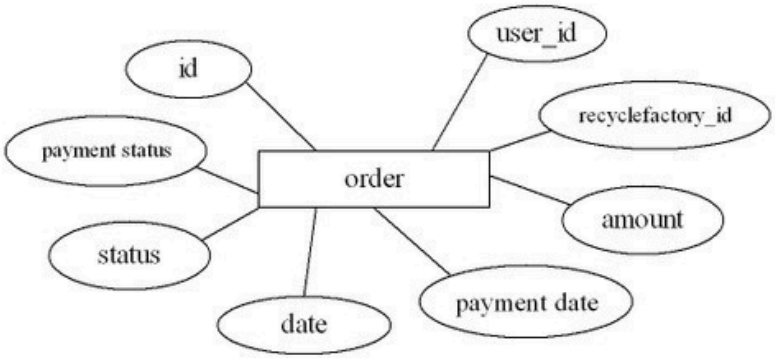


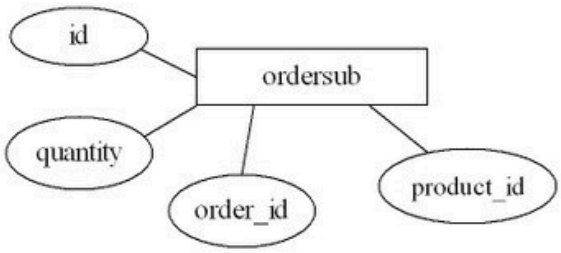


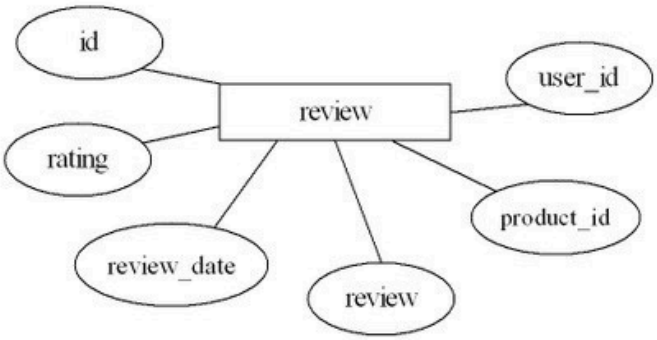





notification
id
date
notification message


user_id
id
recyclefactory_id
payment status
order
amount
status
payment date
date


id
ordersub
quantity
product_id
order_id


id
user_id
review
rating
product_id
review_date
review

5.5 DATABASE DESIGN

1.AREA TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
areaname	varchar	null
place	varchar	null
pin	varchar	null
post	varchar	null
landmark	varchar	null
latitude	varchar	null
longitude	varchar	null
RECYCLEFACTORY_id	int	Foreign key

3.BANK TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
account no	varchar	null
pin	varchar	null
balance	varchar	null
LOGIN_id	int	Foreign key

4.CART TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
quantity	varchar	null
PRODUCT_id	int	Foreign key
USER_id	int	Foreign key

5.CASH TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
paymentdate	varchar	null
payment_type	varchar	null
amount	varchar	null
WASTEDETAILS_id	int	Foreign key

6.CHAT TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
date	varchar	null
message	varchar	null
type	varchar	null
STAFF_id	int	Foreign key
USER_id	int	Foreign key

7.COMPLAINTS TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
complaints	varchar	null
complaintdate	varchar	null
reply	varchar	null
replydate	varchar	null
USER_id	int	Foreign key

8.CREDITPOINTS TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
credit_points	varchar	null
LOGIN_id	int	Foreign key

9.LOGIN TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	primary key
username	varchar	null
password	varchar	null
usertype	varchar	null

10.NOTIFICATION TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
notification_message	varchar	null
date	varchar	null

11.ORDER TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
payment_status	varchar	null
status	varchar	null
date	varchar	null
payment_date	varchar	null
amount	varchar	null
RECYCLEFACTORY_id	int	Foreign key
USER_id	int	Foreign key

12.ORDERSUB TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
quantity	varchar	null
ORDER_id	int	Foreign key
PRODUCT_id	int	Foreign key

13.PRODUCT TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
product_name	varchar	null
category	varchar	null
description	varchar	null
price	varchar	null
stock_quantity	varchar	null
image	varchar	null
RECYCLEFACTORY_id	int	Foreign key

14.RECYCLEFACTORY TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
username	varchar	null
email	varchar	null
phno	varchar	null
place	varchar	null
pin	varchar	null
post	varchar	null
license	varchar	null
LOGIN_id	int	Foreign key

15.REVIEW TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
rating	varchar	null
review_date	varchar	null
review	varchar	null
PRODUCT_id	int	Foreign key
USER_id	int	Foreign key

16.SCRAP_COLLECTION_ENTRY TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
quantity	varchar	null
WASTEDETAILS_id	int	Foreign key

17.STAFF TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
staffname	varchar	null
email	varchar	null
phno	varchar	null
place	varchar	null
pin	varchar	null
post	varchar	null
latitude	varchar	null
longitude	varchar	null
LOGIN_id	int	Foreign key
RECYCLEFACTORY_id	int	Foreign key

18.STAFFALLOCATION TABLE

DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
USER_id	int	Foreign key
STAFF_id	int	Foreign key

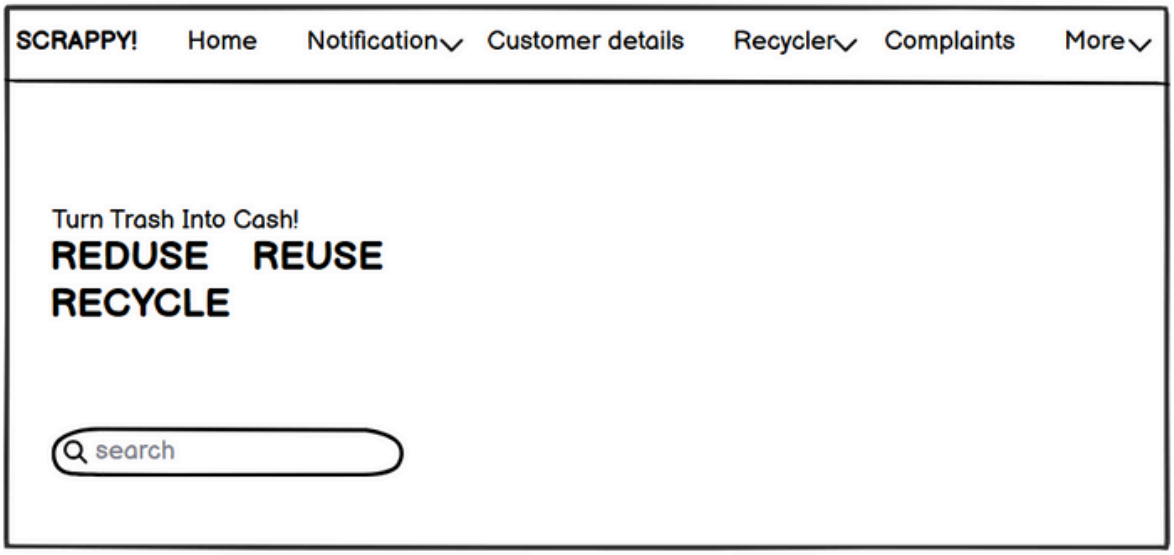
19.USER TABLE

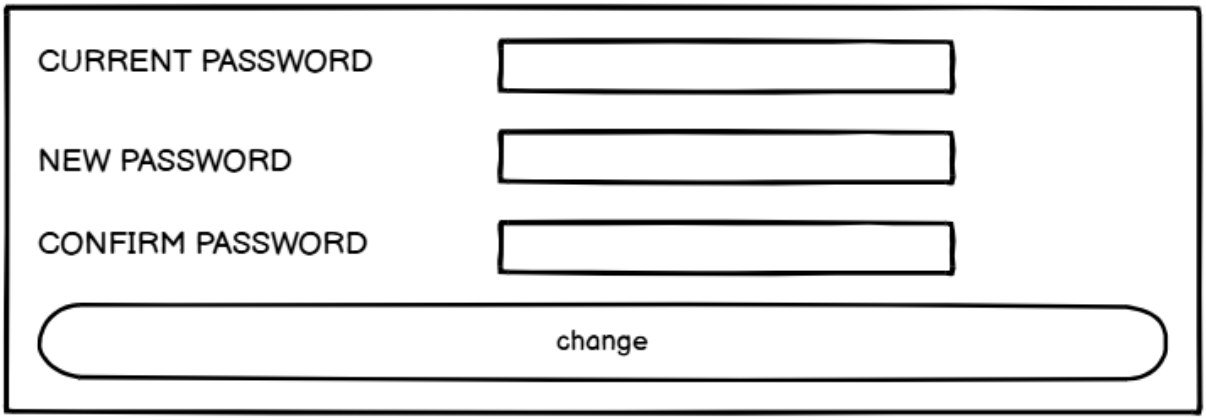
DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
name	varchar	null
phno	varchar	null
email	varchar	null
latitude	varchar	null
longitude	varchar	null
place	varchar	null
pin	varchar	null
post	varchar	null
AREA_id	int	Foreign key
LOGIN_id	int	Foreign key

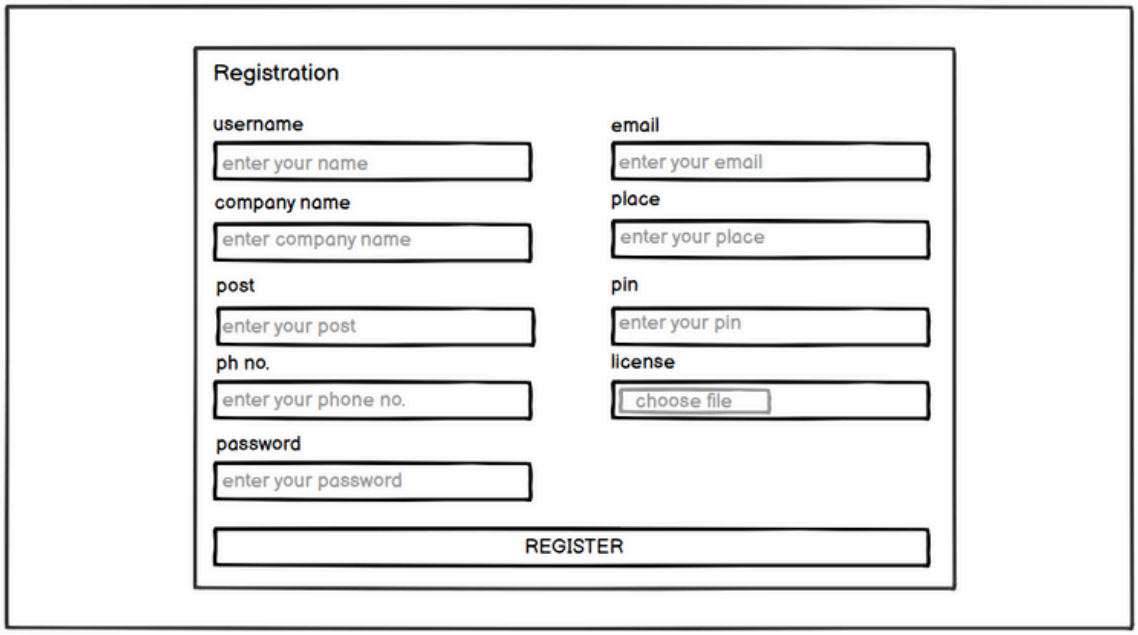
20.WASTEDETAILS TABLE

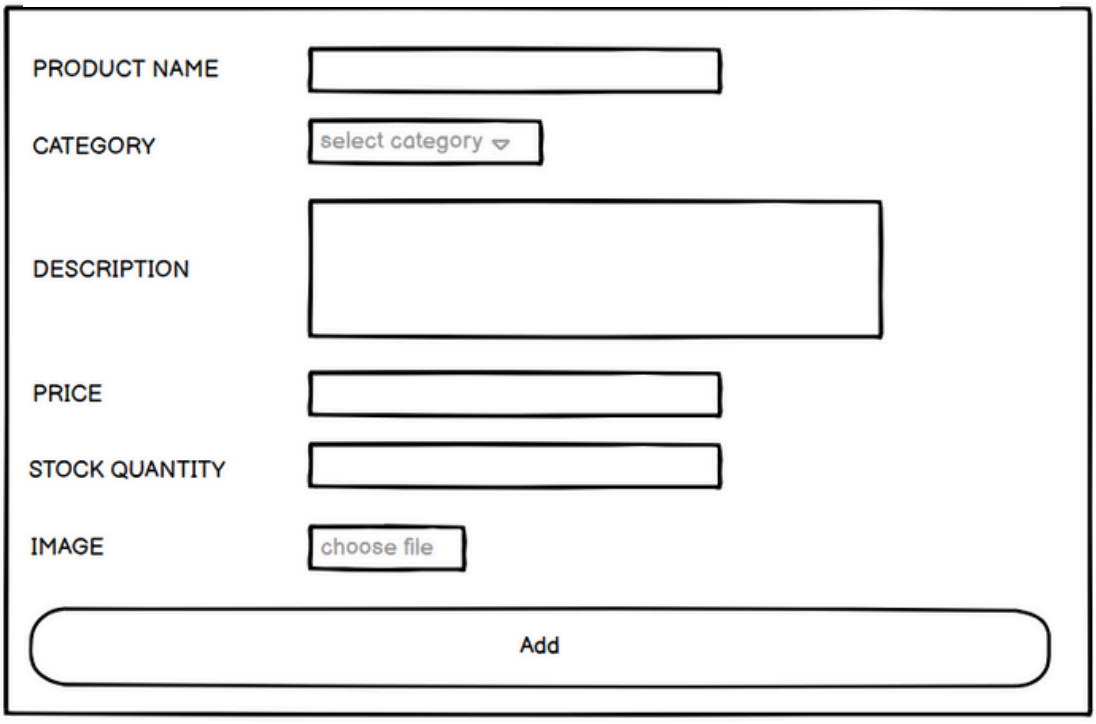
DATA ITEM	DATA TYPE	CONSTRAINTS
id	int	Primary key
waste_type	varchar	null
quantity	varchar	null
date	varchar	null
USER_id	int	Foreign key
STAFF_id	int	Foreign key

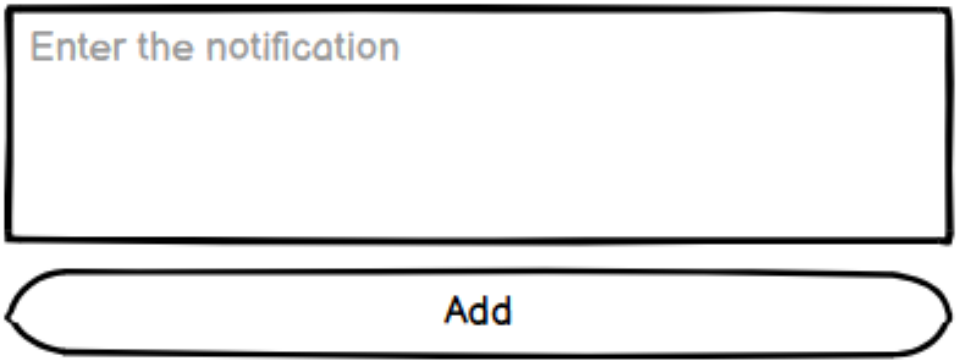
5.6 USERINTERFACE



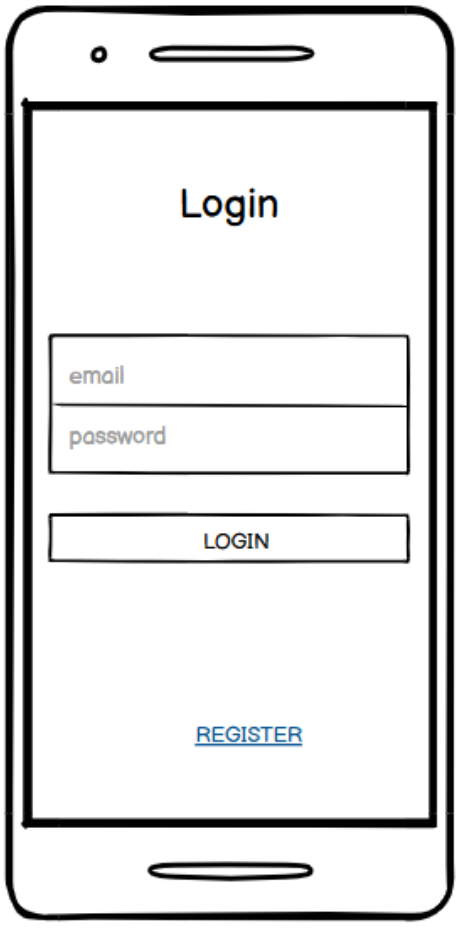


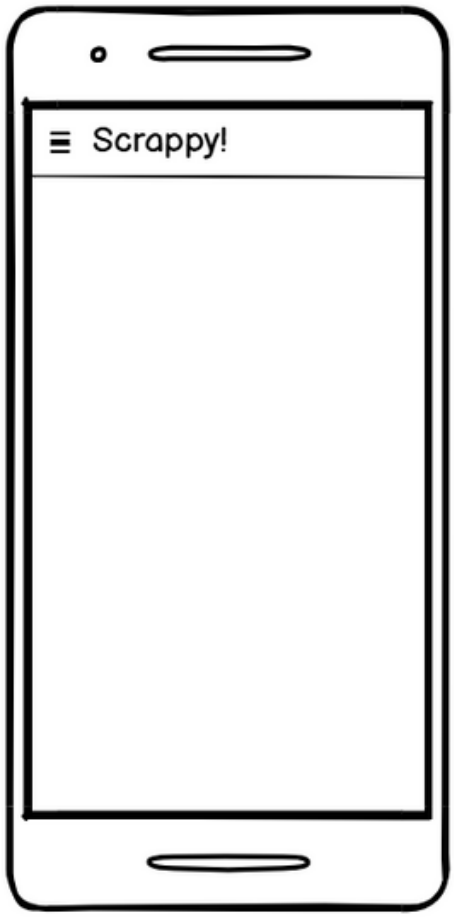

Registration
username
enter your name
email
enter your email
company name
enter company name
place
enter your place
post
enter your post
pin
enter your pin
ph no.
enter your phone no.
license
choose file
password
enter your password
REGISTER

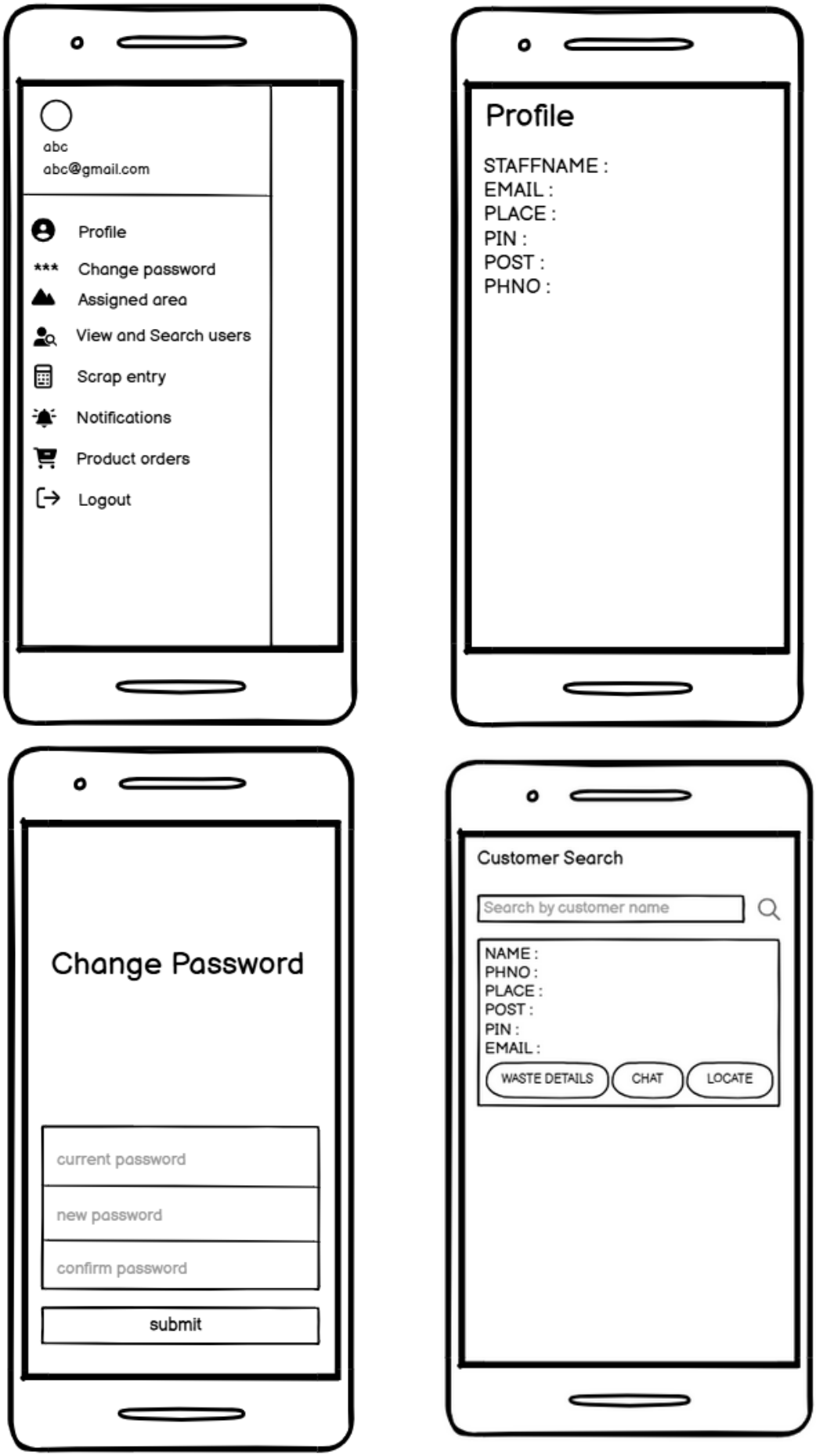

PRODUCT NAME
CATEGORY
select category
DESCRIPTION
PRICE
STOCK QUANTITY
IMAGE
choose file
Add

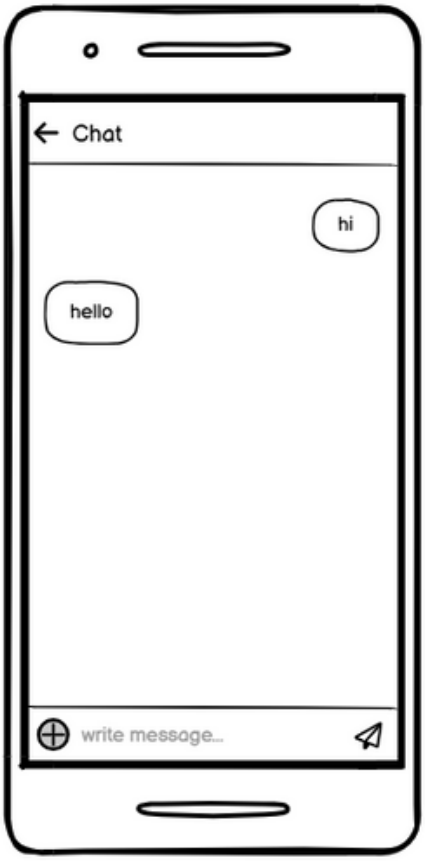


Sl No.	Notification Message	Date

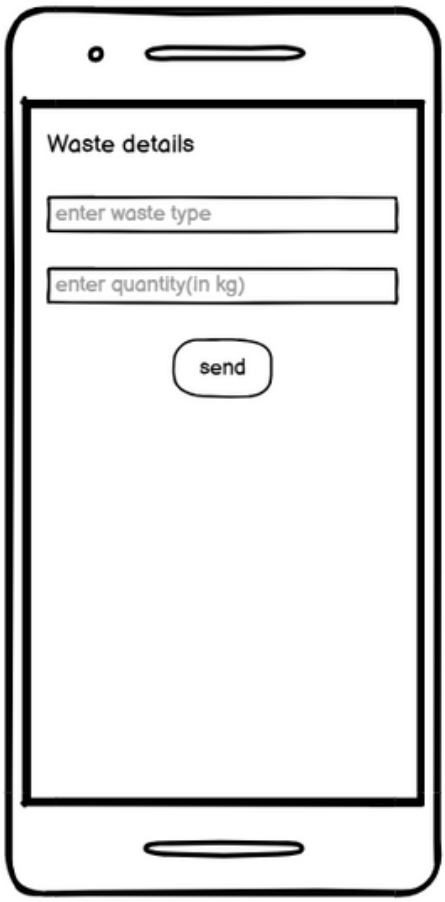




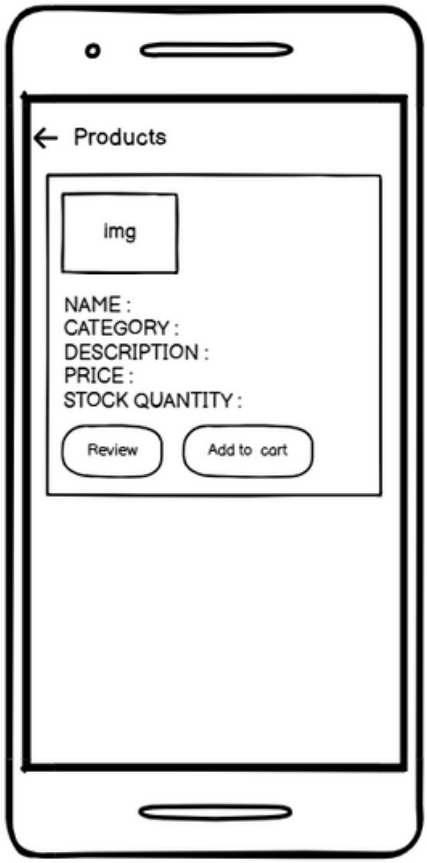

abc
abc@gmail.com
Profile
Change password
Assigned area
View and Search users
Scrap entry
Notifications
Product orders
Logout
Profile
STAFFNAME :
EMAIL :
PLACE :
PIN :
POST :
PHNO :
Change Password
current password
new password
confirm password
submit
Customer Search
Search by customer name
NAME :
PHNO :
PLACE :
POST :
PIN :
EMAIL :
WASTE DETAILS
CHAT
LOCATE


Chat
hi
hello
write message...


REGISTER
username
phone
email
area
latitude
longitude
place
landmark
pin
post
password
confirm password
REGISTER


Waste details
enter waste type
enter quantity(in kg)
send


credit points
credit points available : 0


Products
Img
NAME :
CATEGORY :
DESCRIPTION :
PRICE :
STOCK QUANTITY :
Review
Add to cart

6.CODING

ADMIN

Login :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Untitled Document</title>
</head>
<body><form action="/login_post" method="post"><table width="200" border="1">
<tr>
<th scope="row">username</th>

  <td><label for="textfield"></label>
   <input type="text" name="textfield" id="textfield" /></td>
    </tr>          <tr>          <th
scope="row">password</th>

  <td><label for="textfield2"></label>
   <input type="password" name="textfield2" id="textfield2" /></td>
  </tr>    <th colspan="2" scope="row"><input type="submit" name="button"
id="button" value="Login" /></th>


  </tr>
 <tr>
<th colspan="2" scope="row"><p><a href="/">user </a></p>
   <p><a href="/registration_get">recycle factory </a></p>
{#      <p><a href="/">staff </a></p>#}
   </th>
 </tr>
</table>
</form>
</body>
</html>
```

```
def login_get(request):
return render(request,'index.html')
```

```
def login_post(request):
   un=request.POST['textfield']
 password=request.POST['textfield2']
obj=login.objects.filter(username=un,password=password)
   request.session['login']=1
   if obj.exists():
      lgdata = obj[0]
      request.session['lid'] = lgdata.id
      if lgdata.usertype=='admin':
         return HttpResponse('<script>alert("login
successfully");window.location="/admin_homepage_get"</script>')
      elif lgdata.usertype=='recycler':
         request.session['rid']=recyclefactory.objects.get(LOGIN=lgdata.id).id
         return HttpResponse('<script>alert("login
successfully");window.location="/recycle_factory_homepage_get"</script>')

      elif lgdata.usertype=='staff':
         request.session['lid']=lgdata.id
         request.session['sid']=staff.objects.get(LOGIN=lgdata.id).id
         return HttpResponse('<script>alert("login
successfully");window.location="/staff_home_page_get"</script>')
      else:
         return HttpResponse('<script>alert("invalid details");window.location="/"</script>')
   else:
      return HttpResponse('<script>alert("unauthorised user");window.location="/"</script>')
```

Home page :

```
def              admin_homepage_get(request):
request.session['head']  =  "HOME"     return
render(request,'admin/index.html')
```

Change password :

```
def change_password_get(request):
request.session['head'] = "CHANGE PASSWORD"
return render(request, 'Admin/change password.html')
def change_password_post(request):

   cp=request.POST['textfield']
   np=request.POST['textfield2']
 confirm=request.POST['textfield3']
obj = login.objects.filter(password=cp)
   if obj.exists():
      if np == confirm:
         login.objects.filter(usertype='admin').update(password=confirm)
         return HttpResponse('<script>alert("password successfully");window.location="/"
</script>')
```

```
      else:
        return HttpResponse('<script>alert("password
missmatch");window.location="/change_password_get"</script>')
   else:
      return HttpResponse('<script>alert("old password
missmatch");window.location="/change_password_get"</script>')
```

Notification :

Add:

```
def add_notification_get(request):
   request.session['head'] = "NOTIFICATION"
 return render(request,'admin/add notifications.html')
def add_notification_post(request):

   notifications=request.POST['notification']
 d=datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
i=notification.objects.filter(notification_message=notifications,date=d)
   if i.exists():
      return HttpResponse(
        '<script>alert("notification already
added");window.location="/admin_view_notification_get#aa"</script>')
   else:
      data = notification()
      data.notification_message = notifications
      data.date = d
      data.save()
      return HttpResponse(
        '<script>alert("notification added");window.location="/admin_view_notification_get#aa"
</script>')
```

View :

```
def admin_view_notification_get(request):
request.session['head'] = "VIEW NOTIFICATION"
   obj=notification.objects.all()
 return render(request,'admin/view notification.html',{'view':obj})
```

View customer details :

```
def view_customer_get(request):
request.session['head'] = "VIEW CUSTOMER"
   obj=user.objects.all()
 return render(request,'admin/view customer.html',{'view':obj})
```

View and verifying recyclers :

```
def view_and_verify_recycler_get(request):
request.session['head']="VIEW RECYCLER"
    obj=recyclefactory.objects.filter(LOGIN__usertype='pending')
 return render(request,'admin/view&verify recycler.html',{'view':obj})
def view_verified_recycler_get(request):

    request.session['head'] = "VERIFIED RECYCLER"
    obj = recyclefactory.objects.filter(LOGIN__usertype='recycler')
 return render(request,'admin/view verified recycler.html',{'view':obj})
```

View complaints and send reply :

```
def view_complaints_and_reply_get(request):
request.session['head'] = "VIEW COMPLAINTS"
obj = complaints.objects.all()
return render(request,'admin/view complaints & snd replay.html',{'view':obj})
def send_reply(request,id):

    request.session['head'] = "SEND REPLY"
  return   render(request,'admin/send   reply.html',{'id':id})
def                                  send_reply_post(request,id):
reply=request.POST['textarea']

 d=datetime.datetime.now().strftime("%d-%m-%Y")
complaints.objects.filter(id=id).update(reply=reply,replydate=d)
    return HttpResponse('<script>alert("reply
sended");window.location="/view_complaints_and_reply_get#aa"</script>')
```

RECYCLE FACTORY

Registration :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Untitled Document</title>
</head>
<body>
<form action="/registration_post" method="post" enctype="multipart/form-data"
name="form1" id="form1">
<table width="200" border="1">

    <tr>
      <th scope="row">username</th>
```

```
    <td>
    <input type="text" name="textfield" id="textfield" required class="form control"/></td>
   </tr>
   <tr>
    <th scope="row">email</th>
    <td>
    <input type="text" name="textfield2" id="textfield2" required class="form control"/></td>
   </tr>
    <tr>
    <th scope="row">company_name</th>
    <td>
    <input type="text" name="textfield8" id="textfield8" required class="form control"/></td>
   </tr>
   <tr>
    <th scope="row">place</th>
    <td>
    <input type="text" name="textfield3" id="textfield3" required class="form control"/></td>
   </tr>
   <tr>
    <th scope="row">post</th>
    <td>
    <input type="text" name="textfield4" id="textfield4" required class="form control"/></td>
   </tr>
   <tr>
    <th scope="row">pin</th>
    <td>
    <input type="text" name="textfield5" id="textfield5" required class="form control"/></td>
   </tr>
   <tr>
    <th scope="row">phno</th>
    <td><p>

      <input type="text" name="textfield6" id="textfield6" required class="form control"/>
    </p></td>
   </tr>
   <tr>
    <th scope="row">license</th>
    <td>
    <input type="file" name="fileField" id="fileField" required class="form control"/></td>
   </tr>
   <tr>
    <th scope="row">password</th>
    <td>
    <input type="password" name="textfield7" id="textfield7" required class="form control"/>
</td>
   </tr>
   <tr>
```

```
    <th colspan="2" scope="row"><input type="submit" name="button" id="button"
value="Submit" /></th>
  </tr>
 </table>
</form>
</body>
</html>



def registration_get(request):
return render(request,'register.html')
def registration_post(request):
username=request.POST['textfield']
email=request.POST['textfield2']

  phno=request.POST['textfield6']
 company_name=request.POST['textfield8']
  place=request.POST['textfield3']
  post=request.POST['textfield4']
  pin=request.POST['textfield5']
 password=request.POST['textfield7']
  license = request.FILES['fileField']
 fs=FileSystemStorage()
d=datetime.datetime.now().strftime("%Y%m%d%H%M%S")
fs.save(r"C:\Users\sahalu\PycharmProjects\scrap_collection\media\image\\"+d+
".pdf",license)
license="/media/image/"+d+".pdf"
i=recyclefactory.objects.filter(username=email,email=email,license=license)
  if i.exists():
    return HttpResponse('<script>alert("registered already");window.location="/"</script>')
  else:
    obj1 = login()
    obj1.usertype = "pending"
    obj1.password = password
    obj1.username = email
    obj1.save()
    obj = recyclefactory()
    obj.LOGIN = obj1
    obj.username = username
    obj.email = email
    obj.phno = phno
    obj.company_name=company_name
    obj.place = place
    obj.post = post
    obj.pin = pin
    obj.license = license
    obj.save()
```

```
        obj = credit_points()
        obj.credit_points = 0
        obj.LOGIN_id = obj1.id
        obj.save()
        return HttpResponse('<script>alert("registered successfully");window.location="/"
</script>')
```

Home page :

```
def recycle_factory_homepage_get(request):
return render(request,'recycle factory/index.html')
```

Profile :

```
def view_profile_get(request,):
view=recyclefactory.objects.get(LOGIN=request.session['lid'])
return render(request,'recycle factory/view profile.html',{"view":view})
```

Area :

Add :

```
def add_area_get(request):
return render(request,'recycle factory/add area.html')
def add_area_post(request):
areaname=request.POST['textfield']

    place=request.POST['textfield2']
    pin=request.POST['textfield3']
    post=request.POST['textfield6']
    latitude=request.POST['textfield4']
    longitude=request.POST['textfield5']
 i=area.objects.filter(areaname=areaname)
    if i.exists():
        return HttpResponse('<script>alert("area already
added");window.location="/view_area_get#aa"</script>')
    else:
        obj = area()
        obj.areaname = areaname
        obj.place = place
        obj.pin = pin
        obj.post = post
        obj.latitude = latitude
        obj.longitude = longitude
        obj.RECYCLEFACTORY_id = request.session['rid']
        obj.save()
        return HttpResponse('<script>alert("area added");window.location="/view_area_get#aa"
</script>')
```

View :

```
def view_area_get(request):
   obj=area.objects.all()
 return render(request,'recycle factory/view area.html',{"view":obj})
```

Product :

Add :

```
def add_product_get(request):
return render(request,'recycle factory/add product.html')
def add_product_post(request):
product_name = request.POST['textfield']

   category = request.POST['select']
   description = request.POST['textarea']
   price = request.POST['textfield2']
   stock_quantity= request.POST['textfield3']
 image = request.FILES['fileField']
fs = FileSystemStorage()
d = datetime.datetime.now().strftime("%Y%m%d%H%M%S")
fs.save(r"C:\Users\sahalu\PycharmProjects\scrap_collection\media\productimage\\" + d +
".jpg",
          image)
 path = "/media/productimage/" + d + ".jpg"
i=product.objects.filter(product_name =
product_name,RECYCLEFACTORY=recyclefactory.objects.get(LOGIN=request.session['lid']))
   if i.exists():
      return HttpResponse('<script>alert("product already
added");window.location="/view_product_get#aa"</script>')
   else:
      obj = product()
      obj.product_name = product_name
      obj.category = category
      obj.description = description
      obj.price = price
      obj.stock_quantity = stock_quantity
      obj.image = path
      obj.RECYCLEFACTORY = recyclefactory.objects.get(LOGIN=request.session['lid'])
      obj.save()
      return HttpResponse('<script>alert("product
added");window.location="/view_product_get#aa"</script>')
```

View :

```
def view_product_get(request):
   obj=product.objects.filter(RECYCLEFACTORY__LOGIN_id=request.session['lid'])
 return render(request,'recycle factory/view product.html',{"view":obj})
```

Staff :

Add :

```
def add_staff_get(request):
return render(request,'recycle factory/add staff.html')
def add_staff_post(request):
name=request.POST['textfield']
email=request.POST['textfield2']

   phno=request.POST['textfield3']
   place=request.POST['textfield8']
   pin=request.POST['textfield5']
   post=request.POST['textfield4']
   latitude=request.POST['textfield6']
   longitude=request.POST['textfield7']
 i=login.objects.filter(username=email)
   if i.exists():
      return HttpResponse('<script>alert("staff already
added");window.location="/view_staff_get#aa"</script>')
   else:
      obj1 = login()
      obj1.username = email
      obj1.password = random.randint(0000, 9999)
      obj1.usertype = 'staff'
      obj1.save()
      obj = staff()
      obj.staffname = name
      obj.email = email
      obj.phno = phno
      obj.place = place
      obj.pin = pin
      obj.post = post
      obj.latitude = latitude
      obj.longitude = longitude
      obj.RECYCLEFACTORY_id = request.session['rid']
      obj.LOGIN = obj1
      obj.save()
      return HttpResponse('<script>alert("staff added");window.location="/view_staff_get#aa"


</script>')
```

View :

```
def view_staff_get(request):
    obj = staff.objects.all()
 return render(request, 'recycle factory/view staff.html', {"view": obj})
```

Product orders :

```
def view_product_order_get(request):
    data = order.objects.filter( Q(status="ordered") | Q(status="packed") | Q(status="shipped"),
RECYCLEFACTORY__LOGIN=request.session['lid'])
    # print(obj)
 return render(request,'Recycle Factory/view product order.html',{"data":data})
```

Credit points :

```
def add_credit_points_get(request):
    res=credit_points.objects.filter(LOGIN_id=request.session['lid'])
    if res.exists():
        res=res[0]
        return render(request, 'recycle factory/add creditpoints.html', {"cred" : res.credit_points})
    else:
        return render(request, 'recycle factory/add creditpoints.html', {"cred":"0"})
def convert_to_payments(request,c):

    balance = bank.objects.filter(LOGIN=request.session['lid'])
    b = int(balance[0].balance) + (int(c)*5)
    balance.update(balance=b)
    credit_points.objects.filter(LOGIN_id=request.session['lid']).update(credit_points=0)
    return HttpResponse('<script>alert("Converted successfully to your
account");window.location="/add_credit_points_get#aa"</script>')
```

Staff allocation :

```
def add_staff_allocation_get(request):
    data=staff.objects.all()
    data1=area.objects.all()
  return render(request,'Recycle Factory/add staff allocation.html',{'data':data,'data1':data1})
def add_staff_allocation_post(request):  staff_member=request.POST['select']


    area=request.POST['select1']
 i=staffallocation.objects.filter(STAFF_id =staff_member,AREA_id=area)
    if i.exists():
        return HttpResponse('<script>alert("staff
allocated");window.location="/view_staff_allocation_get#aa"</script>')
    else:
        obj = staffallocation()
        obj.STAFF_id = staff_member
        obj.AREA_id = area
        obj.save()
        return HttpResponse('<script>alert("staff
allocated");window.location="/view_staff_allocation_get#aa"</script>')
```

STAFF

Login :

```
import 'dart:convert';
import 'dart:html';
import 'package:scrapcollectionn/newregister.dart';
import 'package:scrapcollectionn/staff_homepage.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'home.dart';
import 'login.dart';
void main() {
runApp(MaterialApp(
home: login(),
));
}
class login extends StatefulWidget {
@override
_State createState() => _State();
}
class _State extends State<login> {
final txtUSR = new TextEditingController(text: "haleema@gmail.com");
final txtPSW = new TextEditingController(text: "user");
String? errormsg;
bool? error, showprogress;
bool passwordVisible=true;
Future _saveDetails(String usr, String psw) async {
SharedPreferences prefs = await SharedPreferences.getInstance();




   var url = prefs.getString("url");

 var data = await http.post(
      Uri.parse(url.toString()+"/user_login"),
      body: {"username":usr,"password":psw});

   var jsondata = json.decode(data.body);
   print(jsondata["status"]);
   if(jsondata["status"]=="ok"){
      setState(() {
        print("lllllllllllllll");

      });
      //save the data returned from server
      //and navigate to home page
```

```
    String uid = jsondata["lid"].toString();
    String type = jsondata["type"].toString();
    print("jjjjjjjjjj$type");
    SharedPreferences prefs = await SharedPreferences.getInstance();
    prefs.setString("lid",uid);
    if(type == "user"){
    Navigator.push(

        context,
        MaterialPageRoute(
          builder: (context) =>
            home()));
    }
    e l s e   i f ( t y p e = =   " s t a f f " ) {
     Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) =>
            staff_home()));
    }
   }
   e l s e {
 }  print("lllllllllllllllllllllllllll");




}
@override
Widget build(BuildContext context) {
 return Scaffold(

    appBar: AppBar(
      title: Text('Scrap Collection'),


    ),
    body: Padding(

       padding: EdgeInsets.all(15),
       child: Column(
        children: <Widget>[
          Padding(
            padding: EdgeInsets.all(15),
            child: TextField(
```

```
      decoration: InputDecoration(
        border: OutlineInputBorder(),
        labelText: 'Username',
        hintText: 'Enter Your Username',
        prefixIcon: Icon(Icons.email_outlined)
      ),
      controller: txtUSR,
    ),
  ),
  Padding(
    padding: EdgeInsets.all(15),
    child: TextField(

      decoration: InputDecoration(
        border: OutlineInputBorder(),
        labelText: 'Password',
        hintText: 'Enter Your Password',
        prefixIcon: Icon(Icons.email_outlined)
      ),
      controller: txtPSW,
    ),
  ),
  Padding(padding: EdgeInsets.all(20),
    child: SizedBox(
      width: 100,
      height: 30,
      child: ElevatedButton(
        // textColor: Colors.white,
        //     color:     Colors.blue,
        child:          Text('LOGIN'),
        onPressed: (){
          print("k");
          _saveDetails(txtUSR.text.toString(), txtPSW.text.toString());
          // _saveDetails(txtUSR.text.toString(), txtPSW.text.toString());
          },


      ),
    ),

  ),
  Padding(padding: EdgeInsets.all(20),
    child: SizedBox(
      width: 100,
      height: 30,
      child: ElevatedButton(
        // textColor: Colors.white,
        // color: Colors.blue,
```

```
                child: Text('REGISTER'),
                onPressed: (){
                  print("k");
                  // _saveDetails(txtUSR.text.toString(), txtPSW.text.toString());
                  Navigator.push(
                     context,
                     MaterialPageRoute(
                        builder: (context) =>
                           NewRegister()));
                  // _saveDetails(txtUSR.text.toString(), txtPSW.text.toString());
                },

              ),
            ),

           ),
          ],
        )
 } }  )
   );
class Joke1
{   final String id;
Joke1(this.id); }
```

Home page :

```
def staff_home_page_get(request):
return render(request,'staff/index.html')
```

Profile :

```
def staff_view_profile_get(request):

   lid=request.POST['lid']
   print(lid,"l")
 view = staff.objects.get(LOGIN=lid)
return JsonResponse({"status":"ok","staffname":view.staffname, "email":view.email,
"phno":view.phno,
                  "place":view.place, "pin":view.pin, "post":view.post, "latitude":view.latitude,
"longitude":view.longitude})
```

Change password :

```
def staff_change_password_get(request):
return render(request,'staff/change password.html')
def staff_change_password_post(request):
    cp = request.POST['textfield']
    np = request.POST['textfield2']
 confirm = request.POST['textfield3']
obj = login.objects.filter(password=cp, id=request.session['lid'])
    if obj.exists():
        if np == confirm:
            login.objects.filter(id=request.session['lid']).update(password=confirm)
            return JsonResponse({"status":"ok"})
        else:
            return JsonResponse({"status":"ok"})
    else:
        return JsonResponse({"status":"ok"})
```

Assigned area :

```
def view_assigned_area_get(request):
    lid=request.POST['lid']
    obj = staffallocation.objects.filter(STAFF__LOGIN=lid)
    ar = []
    for i in obj:
        ar.append({
        'id':i.id,
        'areaname':i.AREA.areaname,
        'place':i.AREA.place,
        'pin':i.AREA.pin,
        'latitude':i.AREA.latitude,
        'longitude':i.AREA.longitude,
        })
 return JsonResponse({"message":ar})
```

Scrap collection :

```
def view_today_waste_details(request):
    lid=request.POST['lid']
 date = datetime.datetime.now().strftime("%Y-%m-%d")
    res = staffallocation.objects.filter(STAFF__LOGIN=lid)#
    print("ressssss",res)
    ar = []
    for i in res: # fetch each area
        print(i.AREA_id)
        obj = wastedetails.objects.filter(USER__AREA_id=i.AREA_id,date = date)
        print("objjjjjjjjjjjjj",obj)
        for j in obj:
            print(j.date,"oooo")
```

```
 if cash.objects.filter(WASTEDETAILS=j.id).exists():
        pass
    else:
        ar.append({
        "id": j.id, "type": j.waste_type, "quantity": j.quantity, "date": j.date,
            "name":j.USER.name,"phno":j.USER.phno,"place":j.USER.place,"post":j.USER.post,
                "pin":j.USER.pin,"email":j.USER.email, "lati":j.USER.latitude, "longi":j.USER.longitude})

    print("arrrrrrrrrrrrrr",ar)
return JsonResponse({"message": ar})
```

View and search customers :

```
def view_and_search_customer_get(request):
    lid = request.POST['lid']
    print(lid, 'fghdjs')
    res = user.objects.all()
    area_alloc = staffallocation.objects.filter(STAFF__LOGIN_id=lid)
    area_ids = []
    for i in area_alloc:
        area_ids.append(int(i.AREA_id))
    data = []
    for i in res:
        if int(i.AREA_id) in area_ids:
            data.append({
                'id':i.id,
                'name':i.name,
                'phno':i.phno,
                'email':i.email,
                'latitude':i.latitude,
                'longitude':i.longitude,
                'place':i.place,
                'pin':i.pin,
                'post':i.post,
            })
return JsonResponse({"message":data})
def view_and_search_customer_post(request):
cname=request.POST['select']
res=user.objects.filter(name__startswith=cname)

    area_alloc = staffallocation.objects.filter(STAFF__LOGIN_id=request.session['lid'])
    area_ids = []
    for i in area_alloc:
        area_ids.append(int(i.AREA_id))
    data = []
    for i in res:
        if int(i.AREA_id) in area_ids:
            data.append({
```

```
            'id': i.id,
            'name': i.name,
            'phno': i.phno,
            'email': i.email,
            'latitude': i.latitude,
            'longitude': i.longitude,
            'place': i.place,
            'pin': i.pin,
            'post': i.post,
        })
        # print(obj)


 return JsonResponse({"message":data})
def search_customer_get(request):

   return JsonResponse({"status":"ok"})
```

Notification :

```
def view_notification_staff_get(request):
   obj=notification.objects.all()
   ar= []
   for i in obj:
      ar.append({
      'id':i.id,
      'notification_message':i.notification_message,
      'date':i.date,
      })
 return JsonResponse({"message":ar})
```

View product orders :

```
def staff_view_product_order_get(request):

   lid=request.POST['lid']
   print(lid,'dfgh')
   area_alloc=staffallocation.objects.filter(STAFF__LOGIN_id=lid)
   area_ids=[]
   for i in area_alloc:
      area_ids.append(int(i.AREA_id))
   data=[]
   res =
order.objects.filter(Q(status="shipped")|Q(status="out_for_delivery")|Q(status="delivered")).or
der_by("-id")
   for i in res:
      if int(i.USER.AREA_id) in area_ids:
         data.append({
            'payment_status':i.payment_status,
            'status':i.status,
```

```
        'date':i.date,
        'payment_date':i.payment_date,
        'amount':i.amount,
        'id':i.id,
      })
# print(obj)
return JsonResponse({"message":data})
```

Logout :

```
def logout(request):
   request.session['login'] = 0
   return HttpResponse(
      '<script>alert("logout successfully");window.location="/"</script>')
```

USER

Login :

```
def user_login(request):
username = request.POST['username']
password = request.POST['password']
print(username, password)
res=login.objects.filter(username=username, password=password)
   print(res)
   if res.exists():
      res=res[0]
      print(res)
      lid = res.id
      return JsonResponse({"status":"ok","lid":lid,"type":res.usertype})


   else:
      return JsonResponse({"status":"no"})
```

Registration :

```
import 'dart:convert';
import 'package:animate_do/animate_do.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import 'package:http/http.dart' as http;
import 'home.dart';
void main() => runApp(
MaterialApp(
debugShowCheckedModeBanner: false,
home: NewRegister(),
),
);
```

```
class NewRegister extends StatefulWidget {
@override
_NewRegisterState createState() => _NewRegisterState();
}
class _NewRegisterState extends State<NewRegister> {
var res;
String dropdownValue = '';
List<dynamic> actorsName = [];
String holder = '';
// TextEditingControllers for each field
final TextEditingController usernameController = TextEditingController();
final TextEditingController phoneController = TextEditingController();
final TextEditingController emailController = TextEditingController();
final TextEditingController latitudeController = TextEditingController();
final TextEditingController longitudeController = TextEditingController();
final TextEditingController placeController = TextEditingController();
final TextEditingController landmarkController = TextEditingController();
final TextEditingController pinController = TextEditingController();
final TextEditingController postController = TextEditingController();
final TextEditingController passwordController = TextEditingController();
final TextEditingController confirmPasswordController = TextEditingController();
// For handling async API calls
Future h() async {
SharedPreferences prefs = await SharedPreferences.getInstance();




   var url = prefs.getString("url");
 var data = await http.post(Uri.parse('$url/register_area'));
   var jsondata = json.decode(data.body);
   setState(() {

     actorsName = jsondata['m'];
     dropdownValue = actorsName.isNotEmpty ? actorsName[0]["id"].toString() : '';
   });   //   For   drop-down   item
selection                          void
getDropDownItem() {

   setState(() {
     holder = dropdownValue;
   });
 } @override  void
initState()          {
super.initState();
h(); }
```

```
@override
Widget build(BuildContext context) {
return Scaffold(
backgroundColor: Colors.white,
body: SingleChildScrollView(
    child: Column(
     children: <Widget>[
      // Background and Header
      Container(
       height: 400,
       decoration: BoxDecoration(
       image: DecorationImage(
          image: AssetImage('assets/images/background.png'),
          fit: BoxFit.fill,
        ),
       ),
       child: Stack(
        children: <Widget>[
          Positioned(
           left: 30,
           width: 80,
           height: 200,
           child: FadeInUp(
            duration: Duration(seconds: 1),
            child: Container(
              decoration: BoxDecoration(
               image: DecorationImage(
                image: AssetImage('assets/images/light-1.png'),
               ),
              ),
             ),
            ),
           ),
          ),
          Positioned(
          left: 140,
          width: 80,
          height: 150,
          child: FadeInUp(
             duration: Duration(milliseconds: 1200),
             child: Container(
               decoration: BoxDecoration(
                image: DecorationImage(
                 image: AssetImage('assets/images/light-2.png'),
                ),
               ),
              ),
             ),
            ),
          ),
```

```
    Positioned(
     child: FadeInUp(
      duration: Duration(milliseconds: 1600),
      child: Container(
       margin: EdgeInsets.only(top: 50),
       child: Center(
       child: Text(
          "REGISTER",
          style: TextStyle(
           color: Colors.white,
           fontSize: 40,
           fontWeight: FontWeight.bold,
          ),
         ),
        ),
       ),
      ),
     ),
    ],
 ),
),
// Registration Form
Padding(
padding: EdgeInsets.all(30.0),
child: Column(
children: <Widget>[

    FadeInUp(
     duration: Duration(milliseconds: 1800),
     child: Container(
      padding: EdgeInsets.all(5),
      decoration: BoxDecoration(
       color: Colors.white,
       borderRadius: BorderRadius.circular(10),
       border: Border.all(color: Color.fromRGBO(143, 148, 251, 1)),
       boxShadow: [
       BoxShadow(
          color: Color.fromRGBO(143, 148, 251, .2),
          blurRadius: 20.0,
          offset: Offset(0, 10),
         )
       ],
      ),
      child: Column(
       children: <Widget>[
        _buildTextField(usernameController, "Username"),
        _buildTextField(phoneController, "Phone"),
        _buildTextField(emailController, "Email"),
```

```
        _buildTextField(latitudeController, "Latitude"),
        _buildTextField(longitudeController, "Longitude"),
        _buildTextField(placeController, "Place"),
        _buildTextField(landmarkController, "Landmark"),
        _buildTextField(pinController, "Pin"),
        _buildTextField(postController, "Post"),
        _buildTextField(passwordController, "Password"),
        _buildTextField(confirmPasswordController, "Confirm Password"),
      ],
    ),
   ),
  ),
 ),
 SizedBox(height: 30),
 FadeInUp(


  duration: Duration(milliseconds: 1900),
  child: GestureDetector(
   onTap: () async {
    String email = usernameController.text;
    String password = passwordController.text;
    SharedPreferences prefs = await SharedPreferences.getInstance();
    var url = prefs.getString("url");
    var data = await http.post(Uri.parse('$url/user_login'),


       body: {"username": email, "password": password});

    var jsondata = json.decode(data.body);
    if (jsondata["status"] == "ok") {
     String uid = jsondata["lid"].toString();
     prefs.setString("lid", uid);
     Navigator.push(context, MaterialPageRoute(builder: (context) => home()));
    }
   },
   child: Container(
    height: 50,
    decoration: BoxDecoration(
     borderRadius: BorderRadius.circular(10),
     gradient: LinearGradient(
      colors: [
       Color.fromRGBO(143, 148, 251, 1),
       Color.fromRGBO(143, 148, 251, .6),
      ],
     ),
    ),
    child: Center(
    child: Text(
```

```
                    "Register",
                    style: TextStyle(color: Colors.white, fontWeight: FontWeight.bold),
                  ),
                ),
              ),
            ),
          ),
          SizedBox(height: 70),
          FadeInUp(
            duration: Duration(milliseconds: 2000),
            child: GestureDetector(
              onTap: () {
                Navigator.push(
                  context,
                  MaterialPageRoute(builder: (context) => NewRegister()),
                );
              },
              child: Text(
                "Login",
                style: TextStyle(color: Color.fromRGBO(143, 148, 251, 1)),
              ),
            ),
          ),
        ],
      ),
    ),
  ),
 );
} Widget _buildTextField(TextEditingController controller, String hint) {

  return Container(
    padding: EdgeInsets.all(8.0),
    decoration: BoxDecoration(
       border: Border(bottom: BorderSide(color: Color.fromRGBO(143, 148, 251, 1)))
    ),
    child: TextField(
      controller: controller,
      decoration: InputDecoration(
        border: InputBorder.none,
        hintText: hint,
        hintStyle: TextStyle(color: Colors.grey[700]),
      ),
    ),
  );
 }
}
```

```
    def user_registration(request):
      name = request.POST['name']
      phno = request.POST['phone']
       email = request.POST['email']
    latitude = request.POST['latitude']
    longitude = request.POST['longitude']
    place = request.POST['place']
 landmark = request.POST['landmark']
    pin = request.POST['pin']
    post = request.POST['post']
    aid = request.POST['aid']
 password = request.POST['password']
confirm_password = request.POST['confirm_password']
i=login.objects.filter(username = email)
    if i.exists():
        return JsonResponse({"status": "Email Already exists"})
    else:
        if password == confirm_password:
            obj1 = login()
            obj1.usertype = "user"
            obj1.password = password
            obj1.username = email
            obj1.save()
            obj = user()
            obj.name = name
            obj.phno = phno
            obj.email = email
            obj.latitude = latitude
            obj.longitude = longitude
            obj.place = place
            obj.landmark = landmark
            obj.pin = pin
            obj.post = post
            obj.AREA_id = aid
            obj.LOGIN = obj1
            obj.save()
            return JsonResponse({"status": "ok","lid":obj1.id})

        return JsonResponse({"status": "Password Mismatch"})
```

Profile :

```
def user_view_profile(request):
    lid=request.POST['lid']
    obj=user.objects.get(LOGIN=lid)
    print("objjjjjjj",obj)
    return JsonResponse({"status":"ok",
                         "name": obj.name,
                         "phno": obj.phno,
                         "email": obj.email,
                         "latitude": obj.latitude,
                         "longitude": obj.longitude,
                         "place": obj.place,
                         "landmark":obj.landmark,
                         "pin": obj.pin,
                         "post": obj.post,
                         })
```

Change password :

```
def change_password(request,id):
    lid = id
    print("lidd",lid)
 current_password = request.POST['current_password']
new_password = request.POST['new_password']
confirm_password = request.POST['confirm_password']
obj = login.objects.filter(password=current_password, id=lid)
    if obj.exists():
        if new_password == confirm_password:
            login.objects.filter(id=lid).update(password=confirm_password)
            return JsonResponse({"status":"ok"})
        else:
            return JsonResponse({"status":"no"})
    else:
        return JsonResponse({"status":"noo"})
```

Enter waste details :

```
def enter_waste_details(request,id):
waste_type=request.POST['wastetype']
    quantity=request.POST['quantity']
 print(waste_type,quantity)
obj = wastedetails()
    obj.date = "pending"
 obj.waste_type = waste_type
    obj.quantity = quantity
    obj.USER = user.objects.get(LOGIN=id)
    obj.STAFF_id =
staffallocation.objects.get(AREA=user.objects.get(LOGIN=id).AREA_id).STAFF_id
    obj.save()
    return JsonResponse({"status": "ok"})
```

Send complaints and view reply :

```
def send_complaints_and_view_reply(request,id):
    lid = id
    print(lid,"jhgfds")
 complaint =request.POST['complaint']
obj = complaints()
obj.complaints = complaint
obj.complaintdate = datetime.datetime.now().strftime("%Y-%m-%d")
    obj.reply = "pending"
    obj.replydate = "pending"
    obj.USER=user.objects.get(LOGIN=lid)
    obj.save()
    return JsonResponse({"status":"ok"})
def view_reply(request):
    lid=request.POST['lid']
    print(lid,"hgggggg")
 obj=complaints.objects.filter(USER__LOGIN=lid)
    ar = []
    for i in obj:
        ar.append(
            {
                "id":i.id,
                "complaint":i.complaints,
                "complaint_date":i.complaintdate,
                "reply":i.reply,
                "reply_date":i.replydate,
            }
        )
    print("arrrrrrrrrr",ar)
 return JsonResponse({"message":ar})
```

view cart :

```
def view_cart(request):
    lid = request.POST['lid']
    obj=cart.objects.filter(USER__LOGIN=lid)
    ar = []
    for i in obj:
        total_price = int(i.PRODUCT.price)*int(i.quantity)
        ar.append(
            {
                "id":i.id,
                "quantity":i.quantity,
                "total_price":total_price,
                 "product_name":i.PRODUCT.product_name,
            }
        )
    print("shadha",ar)
 return JsonResponse({"message":ar})
```

View credit points :

```
def view_credit_points(request):
    lid=request.POST['lid']
    obj=credit_points.objects.filter(LOGIN_id=lid)
    if obj.exists():
        return JsonResponse({"credit":obj[0].credit_points})
    else:
        return JsonResponse({"credit":"0"})
```

Notifications :

```
def view_notification(request):
    obj = notification.objects.all()
    ar = []
    for i in obj:
        ar.append(
            {
                "id":i.id,
                "message":i.notification_message,
                "date":i.date,
            }
        )
    print(ar)
    return JsonResponse({"message":ar})
```

Order status :

```
def view_order_status(request):
    lid=request.POST['lid']
    # pid=request.POST['pid']
    obj=order.objects.filter(~Q(status="delivered"),USER__LOGIN=lid)
    ar = []
    for i in obj:
        ar.append(
            {
                "id":i.id,
                "payment_date":i.payment_date,
                "payment_status":i.payment_status,
                "date":i.date,
                "status":i.status,
                "amount":i.amount,
            }
        )
    print(ar,"sgf")
    return JsonResponse({"message":ar})
```

Previous orders :

```
def view_previous_order(request):
    lid=request.POST['lid']
    print("LL ", lid)
  print(datetime.datetime.now().date(),"ddddddddddddd")
    obj=order.objects.filter(USER__LOGIN=lid,status="delivered")
    ar = []
    for i in obj:
        ar.append(
            {
                "id":i.id,
                "shopid":i.RECYCLEFACTORY.id,
                "amount":i.amount,
                "date":i.date,
                "payment_date":i.payment_date,
                "payment_status":i.payment_status,

            }
        )
    print("anuhal",ar)
  return JsonResponse({"message":ar})
```

View products :

```
 def view_product_from_recycler(request):
     pid = request.POST['pid']
     # obj = order.objects.filter(PRODUCT_id = pid)
     obj = product.objects.all()
     ar= []
     for i in obj:
         ar.append(
             {
                 "id":i.id,
                 "name":i.product_name,
                 "category":i.category,
                 "description":i.description,
                 "price":i.price,
                 "stock_quantity":i.stock_quantity,
                 "image":i.image,
             }
         )
     print("ressssssssssssssss",ar)
   return JsonResponse({"message":ar})
```

Chat with staff :

```
def user_sendchat(request):
from_id=request.POST['from_id']
print(from_id,"kkjj")
msg=request.POST['message']
usr = user.objects.get(LOGIN=from_id)
    stf_all = staffallocation.objects.filter(AREA=usr.AREA)
    if stf_all.exists():
        stf_all = stf_all[0]
        from datetime import datetime
        c=chat()
        c.USER=user.objects.get(LOGIN=from_id)
        c.STAFF=stf_all.STAFF
        c.type="user"
        c.message=msg
        c.date=datetime.now().date()
        c.save()
        return JsonResponse({'status':"ok"})
    else:
        return JsonResponse({'status': "no"})


def user_viewchat(request):
from_id=request.POST['from_id']
print(from_id,"frmmmmmmmmm")
usr=user.objects.get(LOGIN=from_id)
    stf_all=staffallocation.objects.filter(AREA=usr.AREA)
    if stf_all.exists():
stf_all=stf_all[0]
print(stf_all)
l = []
data = chat.objects.filter(STAFF=stf_all.STAFF,
USER=user.objects.get(LOGIN=from_id)).order_by('id')
        print(data)
        for res in data:
            l.append({'id': res.id, 'type': res.type, 'msg': res.message, 'date': res.date})
        print(l,"dhsssssssssssss")
        return JsonResponse({'status': "ok", 'data': l})

    else:
        return JsonResponse({'status':"no"})
```

7.TESTING

7.1 TEST CASES

The application underwent manual testing, where each user interface was subjected to tests with both valid and invalid inputs to verify the accuracy of the application's output and error messages. The application demonstrated effective responsiveness to various input types, producing correct outputs and appropriate error messages for different scenarios.

LOGIN

In the login scenario, we use the password and username as inputs. If both inputs are valid, the login is successful, allowing the user to access the home page. However, if either the username or password is invalid, or if one of them is missing, the login will be unsuccessful.

RECYCLER REGISTRATION

In recycler registration, personal details, company details , license , are required fields. Successful registration occurs when all fields are filled and valid. However, if one or ,more fields are missing , or if the name is not in alphabets and the phone number is below ten digits , an error message will be displayed , and registration will not proceed. Additionally , the password and confirm password must match , otherwise , an error message will be shown.

USER REGISTRATION

In user registration, personal details are required fields. Successful registration occurs when all fields are filled and valid. However, if one or more fields are missing, or if the name is not in alphabets and the phone number is below ten digits, an error message wil be displayed, and registration will not proceed. Additionally, the password and confirm password must match; otherwise, an error message will be shown. It will result in unsuccessful registration.

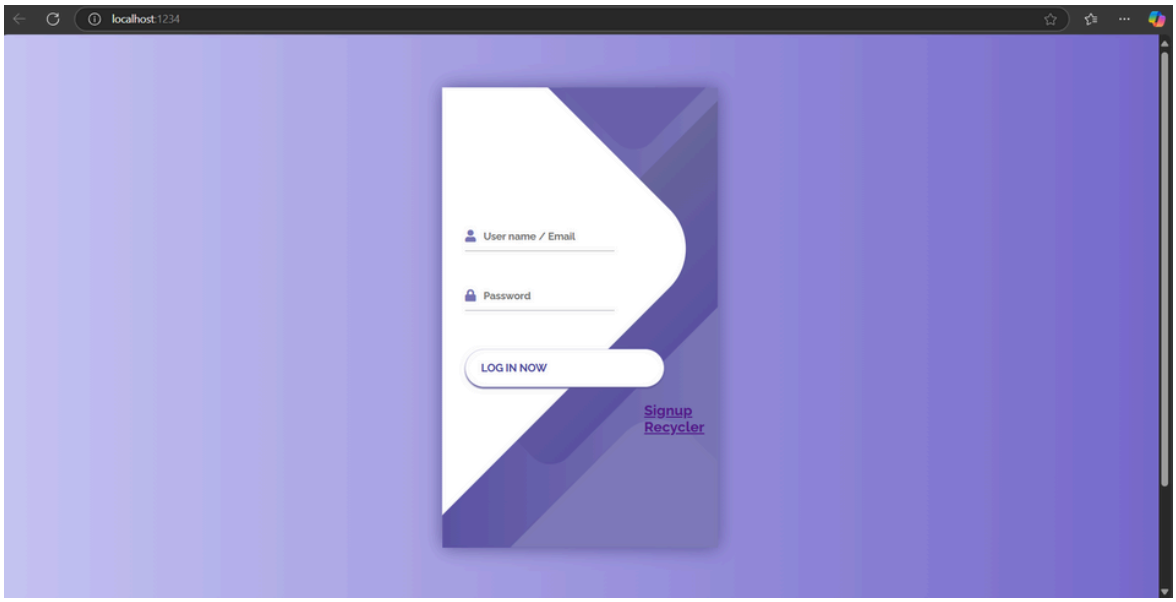
STAFF

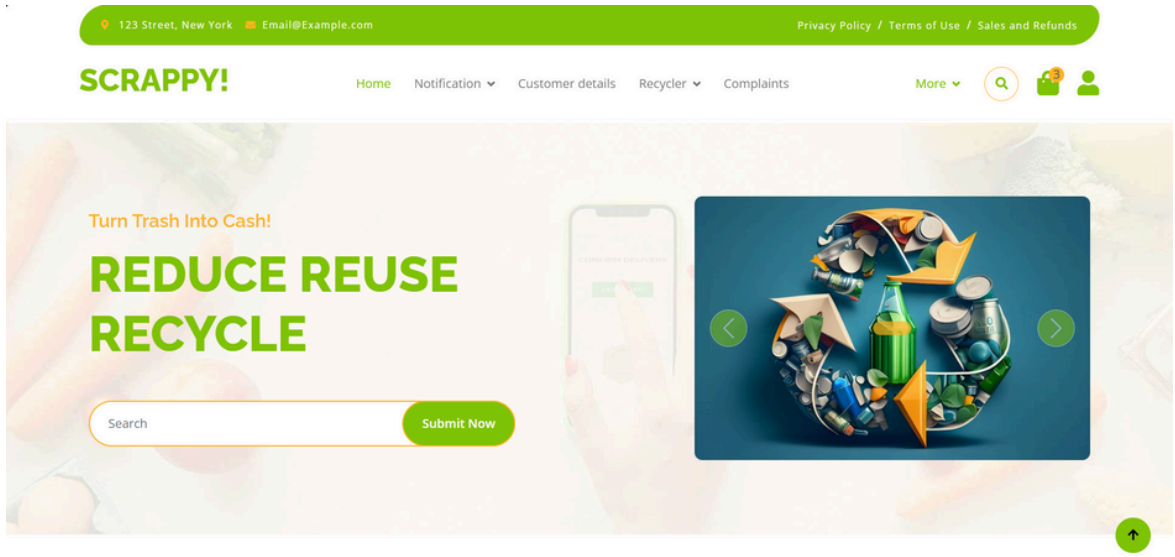
Staff is added by recyclers with necessary details . Staff have the login scenario which uses username and password as its input , allowing the staff to access the home page.

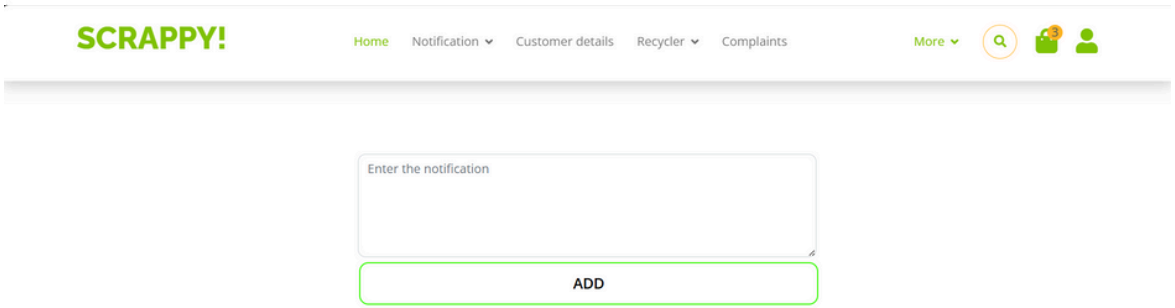
7.2 TEST RESULTS

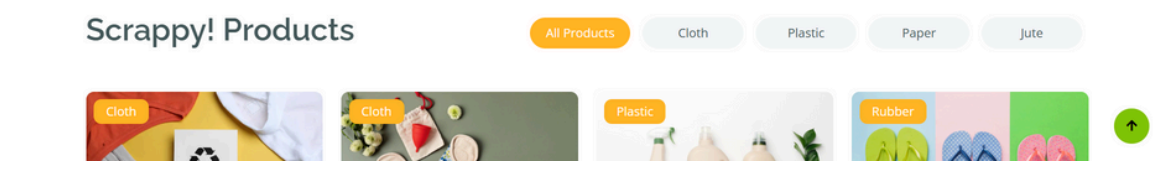
SL.NO	FIELDS	VALID/INVALID	EXPECTED RESULT	REMARKS
1. Login	username and password	valid invalid	login successful invalid username and password	success error
2. Recycler registration	personal details	valid invalid	registration successful invalid entry	success error
3. User registration	personal details	valid invalid	registration successful invalid entry	success error
4. Staff	username and password	valid invalid	login successful invalid username and password	success error

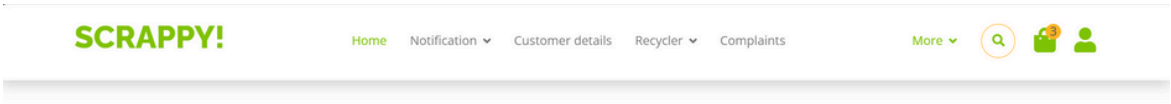
8. SCREENSHOTS

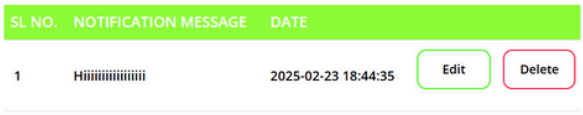


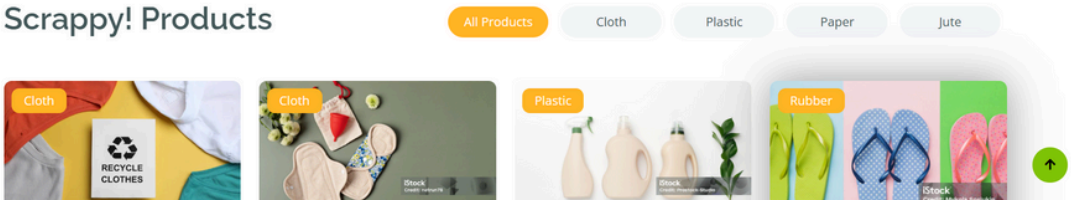


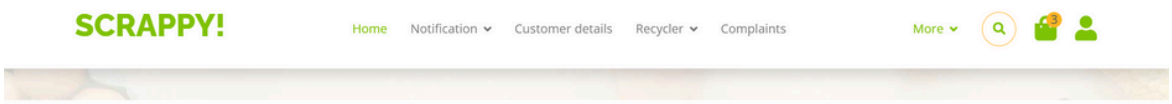





SCRAPPY!
Home
Notification
Customer details
Recycler
Complaints
More

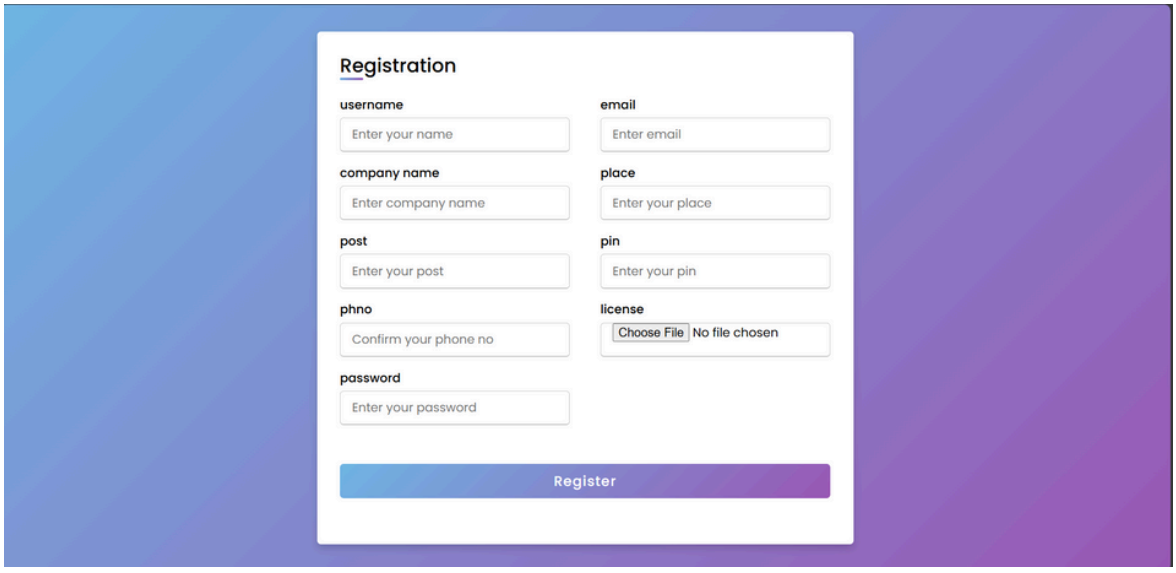

SL NO.
NOTIFICATION MESSAGE
DATE
1
Hiiiiiiiiiiiiiiii
2025-02-23 18:44:35
Edit
Delete

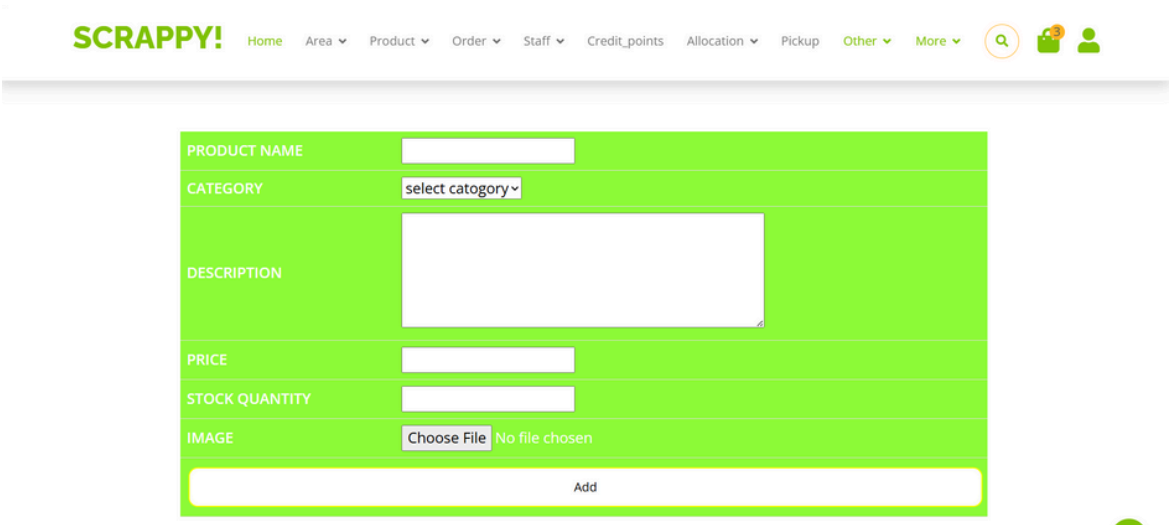

Scrappy! Products
All Products
Cloth
Plastic
Paper
Jute
Cloth
RECYCLE CLOTHES
Cloth
Plastic
Rubber

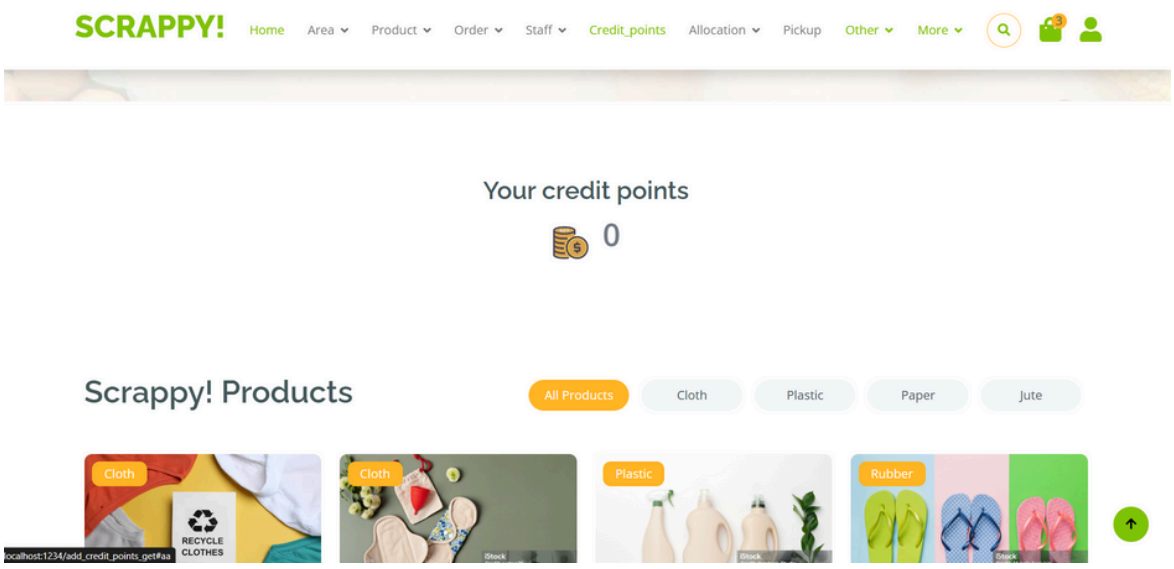

SCRAPPY!
Home
Notification
Customer details
Recycler
Complaints
More


CURRENT PASSWORD
NEW PASSWORD
CONFIRM PASSWORD
change

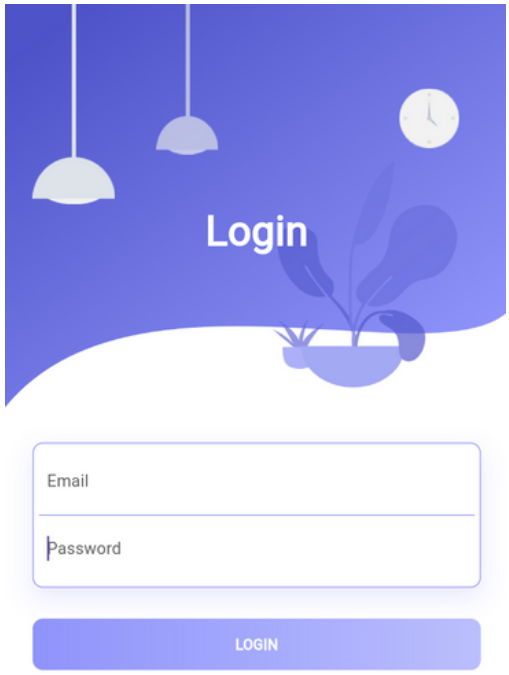

Scrappy! Products
All Products
Cloth
Plastic
Paper
Jute
Cloth
Cloth
Plastic
Rubber

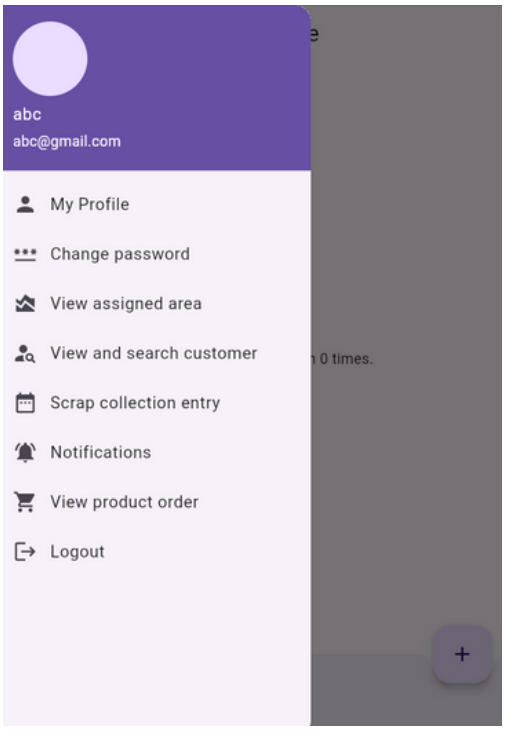

Registration
username
Enter your name
email
Enter email
company name
Enter company name
place
Enter your place
post
Enter your post
pin
Enter your pin
phno
Confirm your phone no
license
Choose File
No file chosen
password
Enter your password
Register

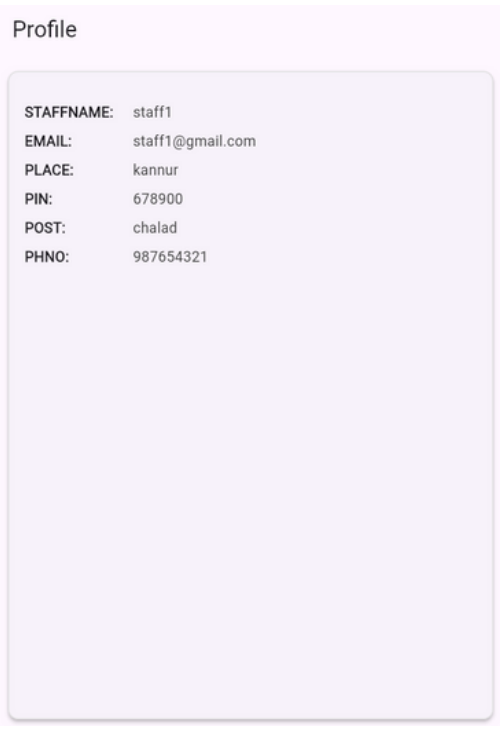

SCRAPPY!
Home
Area
Product
Order
Staff
Credit_points
Allocation
Pickup
Other
More
PRODUCT NAME
CATEGORY
select catogory
DESCRIPTION
PRICE
STOCK QUANTITY
IMAGE
Choose File
No file chosen
Add

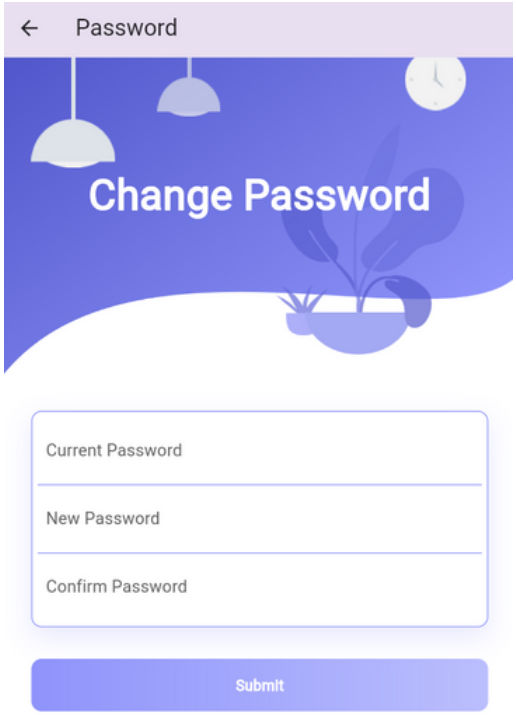

SCRAPPY!
Home
Area
Product
Order
Staff
Credit_points
Allocation
Pickup
Other
More
Your credit points
0
Scrappy! Products
All Products
Cloth
Plastic
Paper
Jute
Cloth
RECYCLE CLOTHES
Cloth
Plastic
Rubber

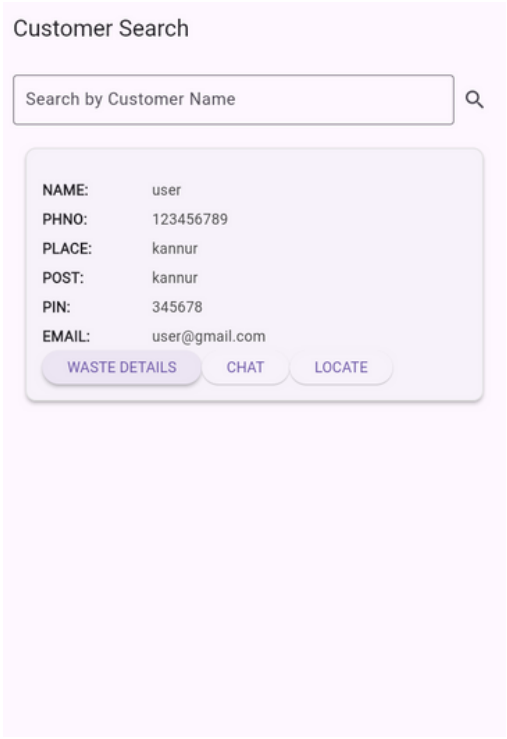

IP PAGE
192.168.70.199
START

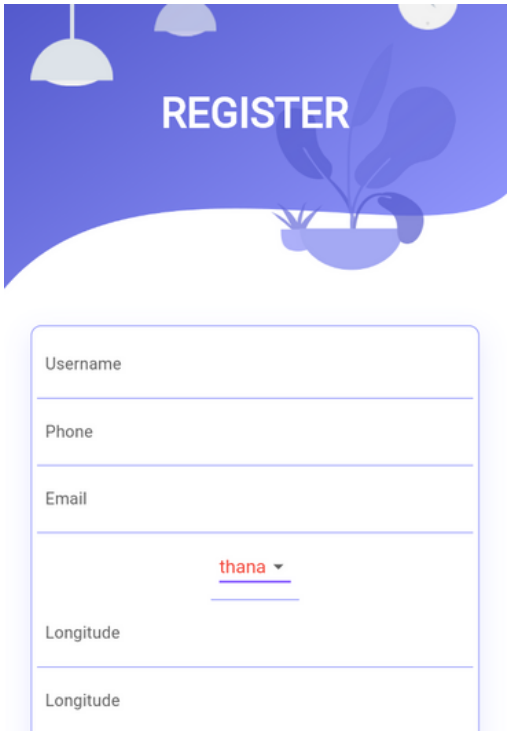

Login
Email
Password
LOGIN

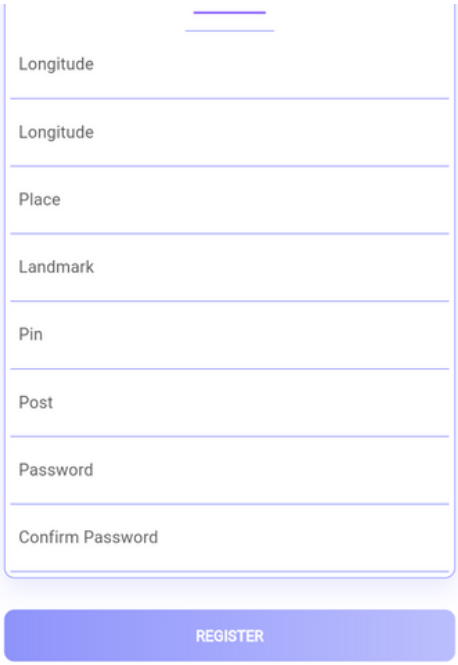

abc
abc@gmail.com
My Profile
Change password
View assigned area
View and search customer
Scrap collection entry
Notifications
View product order
Logout


Profile
STAFFNAME: staff1
EMAIL: staff1@gmail.com
PLACE: kannur
PIN: 678900
POST: chalad
PHNO: 987654321


Password
Change Password
Current Password
New Password
Confirm Password
Submit


Customer Search
Search by Customer Name
NAME: user
PHNO: 123456789
PLACE: kannur
POST: kannur
PIN: 345678
EMAIL: user@gmail.com
WASTE DETAILS
CHAT
LOCATE


REGISTER
Username
Phone
Email
thana
Longitude
Longitude


Longitude
Longitude
Place
Landmark
Pin
Post
Password
Confirm Password
REGISTER

Waste Details

Enter waste type

Enter quantity(in kg)

SEND

Credit points

Credit points available : 70

NAME: bag
CATEGORY: cloth
DESCRIPTION: cloth bag
PRICE: 35
STOCK QUANTITY: 3

Review Add To Cart

9. FUTURE SCOPE

In the future, we will focus on green environment and recycling waste materials to meet market needs. By utilizing GPS technology, we will track users efficiently. Its scope extends beyond facilitating scrap collection, to educating users about proper waste segregation, providing market-based pricing for scrap materials, and connecting consumers with local recycling centers. . Additionally, we will transition our website into a user-friendly application to accommodate the increasing trend in mobile usage.

10. CONCLUSION

In conclusion, we have successfully completed this project. Scrappy now serves as a platform where newcomers can connect with local recyclers, for selling scrap materials. As we move forward, this application has the potential to make a significant impact, inspiring individuals, communities, and industries to embrace eco-friendly practices and strive for a waste-free future."

11. BIBILIOGRAPHY

Software Engineering-KK Agarwal

www.geeksforgeeks.org/python-tutorial

12. GLOSSARY

UML : Unified Modeling Language

DFD : Data Flow Diagram

ERD : Entity Relationship Diagram