CHAPTER 1
INTRODUCTION

1.1 Aim of the Project

Development of waste management system is to make the public friendly access which consumes less time to get BBMP notified related to manage waste and provides a hygiene place.

1.2 Overview of the Project

Waste management system is a web-based application. Through login and sign in page admin, and public can use this application that contains queries regarding place and details can be uploaded through the image also. Thereby reduces the time and provides well management of waste is public places

1.3 Outcome of the Project

By this Project the Student is able to build a Dynamic Web Project using Django server and will have the skills to implement SQL queries dynamically by connecting front end and back end with the help of Servers. The application of our project 'Waste Management System' is like any other conventional management system i.e., we can store the details of the queries regarding waste and only admin will have the access to check the detail and the queries registered on those particular dates users can also view a notified data Our project can be implemented in daily life since waste is commonly found in public place.

1.4 Requirements

Software Configuration

1. OS: Linux, windows
2. Python 3.9.1 (Django Framework)
3. User Interface Design: HTML, CSS, Bootstrap
4. Web Browser: Mozilla, Google Chrome, Opera
5. Software: PyCharm

Hardware Configuration

1. Processor: Any processor above 500 MHz.
2. RAM: 4GB.
3. Hard Disk: 100GB free space.
4. system type: 32-bit or 64-bit operating system.

CHAPTER 2

DESIGN

2.1 Schema Diagram

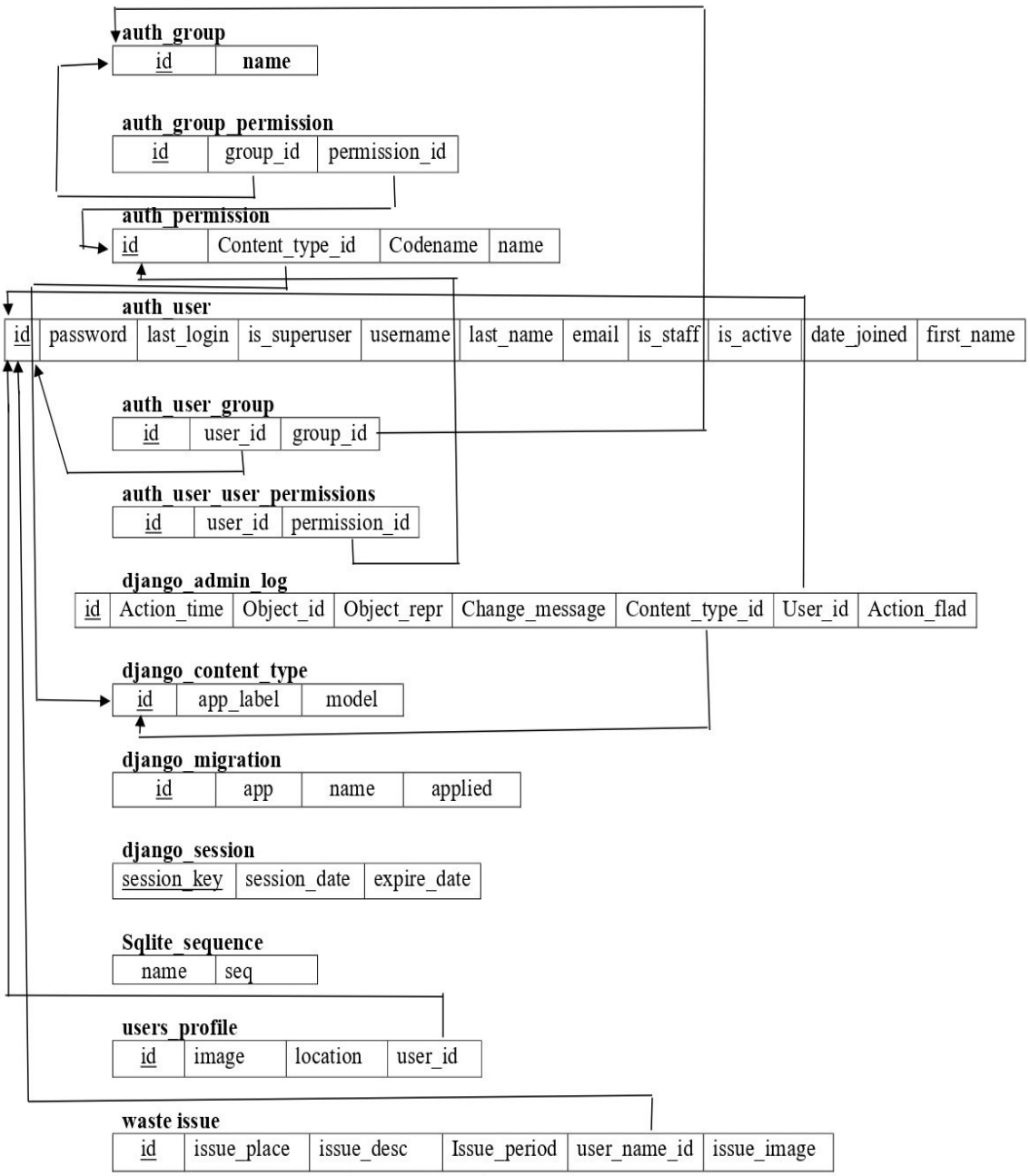

Figure 2.1 Schema Diagram

2.2 ER Diagram

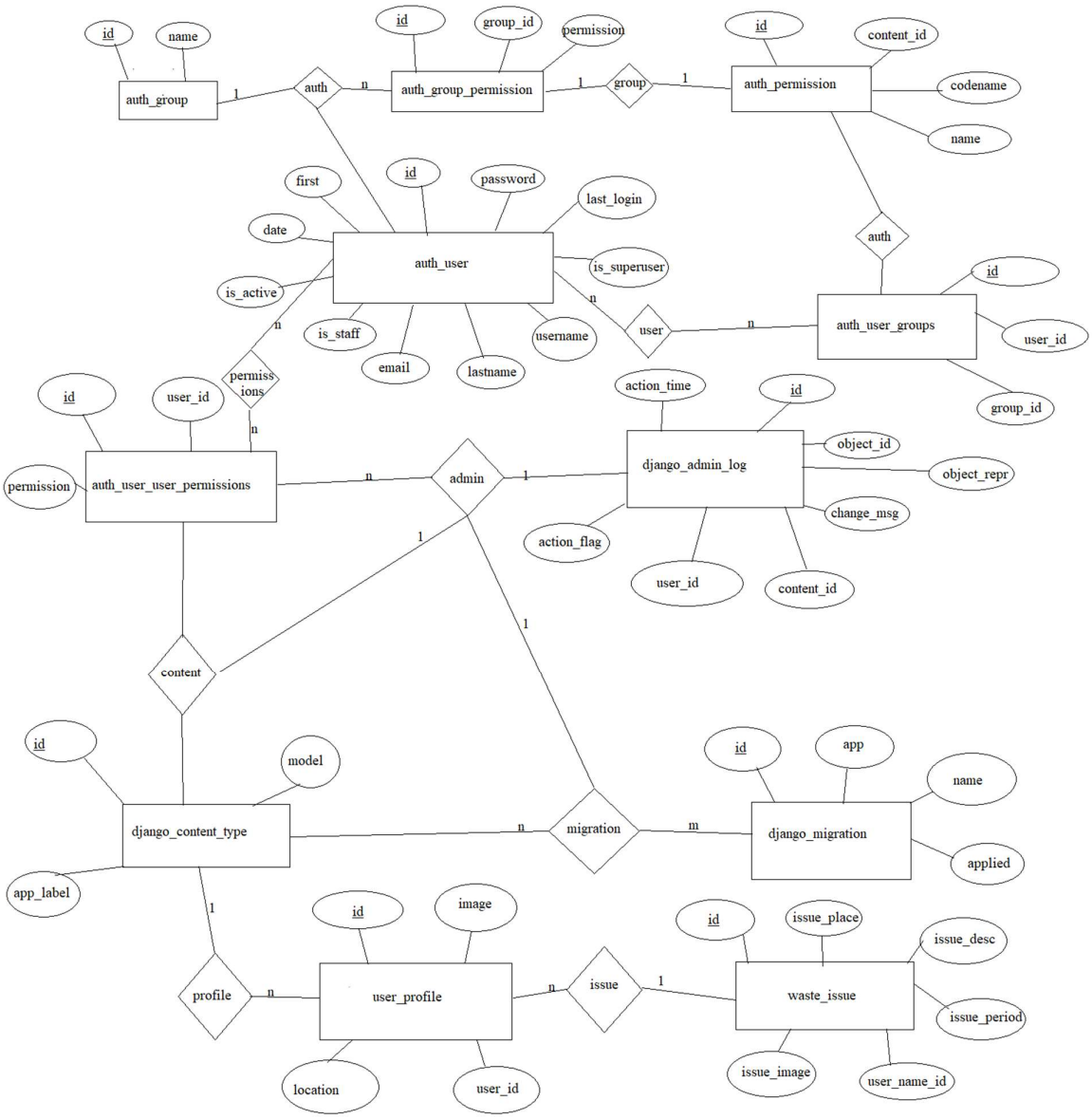

Figure 2.2 ER Diagram

CHAPTER 3
IMPLEMENTATION

3.1 Code Implementation

3.1.1 Models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.urls import reverse

class Issue(models.Model):
def __str__(self):
return self.issue_place
user_name = models.ForeignKey(User,on_delete=models.CASCADE,default=1)
issue_place = models.CharField(max_length=200)
issue_desc = models.CharField(max_length=200)
issue_period = models.IntegerField()
issue_image =
models.CharField(max_length=500,default="https://www.economist.com/img/b/1280/720/90
/sites/default/files/images/2019/06/articles/main/20190622_std001.jpg")

def get_absolute_url(self):
return reverse("waste:detail", kwargs={"pk": self.pk})
```

3.1.2 View.py

```
from django.shortcuts import render,redirect
from django.http import HttpResponse
from .models import Issue
from .forms import IssueForm
from django.contrib.auth.decorators import login_required
from django.views.generic.list import ListView
from django.views.generic.detail import DetailView
def index(request):
    issue_list=Issue.objects.all()
```

```
    context={
        'issue_list':issue_list,
    }
    return render(request,'waste/index.html',context)

class IndexClassView(ListView):
    model = Issue;
    template_name = 'waste/index.html'
    context_object_name = 'issue_list'

def issue(request):
    return HttpResponse('<h1>  this is heading</h1>')

def detail(request,issue_id):
    issue=Issue.objects.get(pk=issue_id)
    context = {
        'issue':issue,
    }
    return render(request,'waste/detail.html',context)

class WasteDetail(DetailView):
    model = Issue;
    template_name = 'waste/detail.html'

def create_issue(request):
    form = IssueForm(request.POST or None)
    if form.is_valid():
        form.save()
        return redirect('waste:index')
    return render(request,'waste/issue-form.html',{'form':form})

def update_issue(request,id):
    issue = Issue.objects.get(id=id)
    form = IssueForm(request.POST or None, instance=issue)
```

```
    if form.is_valid():
        form.save()
        return redirect('waste:index')
    return render(request,'waste/issue-form.html',{'form':form,'issue':issue})


@login_required
def delete_issue(request,id):
    issue = Issue.objects.get(id=id)
    if request.method == 'POST':
        issue.delete()
        return redirect('waste:index')
    return render(request, 'waste/issue-delete.html', { 'issue': issue})
```

3.1.3 Setting.py

```
"""Django settings for mysite project.

Generated by 'django-admin startproject' using Django 3.1.4.

For more information on this file, see

https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see

https://docs.djangoproject.com/en/3.1/ref/settings/

"""

import os

from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.

BASE_DIR = Path(__file__).resolve().parent.parent

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY = '#lybe^n94+)+c1-!4wpn(!p-7@av2329#wd1l0$k#ky%$wdywf'
```

```
# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'waste.apps.WasteConfig',
    'users.apps.UsersConfig',
]

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

ROOT_URLCONF = 'mysite.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
WSGI_APPLICATION = 'mysite.wsgi.application'
# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
}
# Password validation
```

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

AUTH_PASSWORD_VALIDATORS = [

    {

        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',

    },

]

STATIC_URL = '/static/'

LOGIN_REDIRECT_URL = 'waste:index'

LOGIN_URL = 'login'

MEDIA_ROOT = os.path.join(BASE_DIR,'pictures')

MEDIA_URL = '/pictures/'

                    }

            }

}
```

CHAPTER 4

SNAPSHOTS

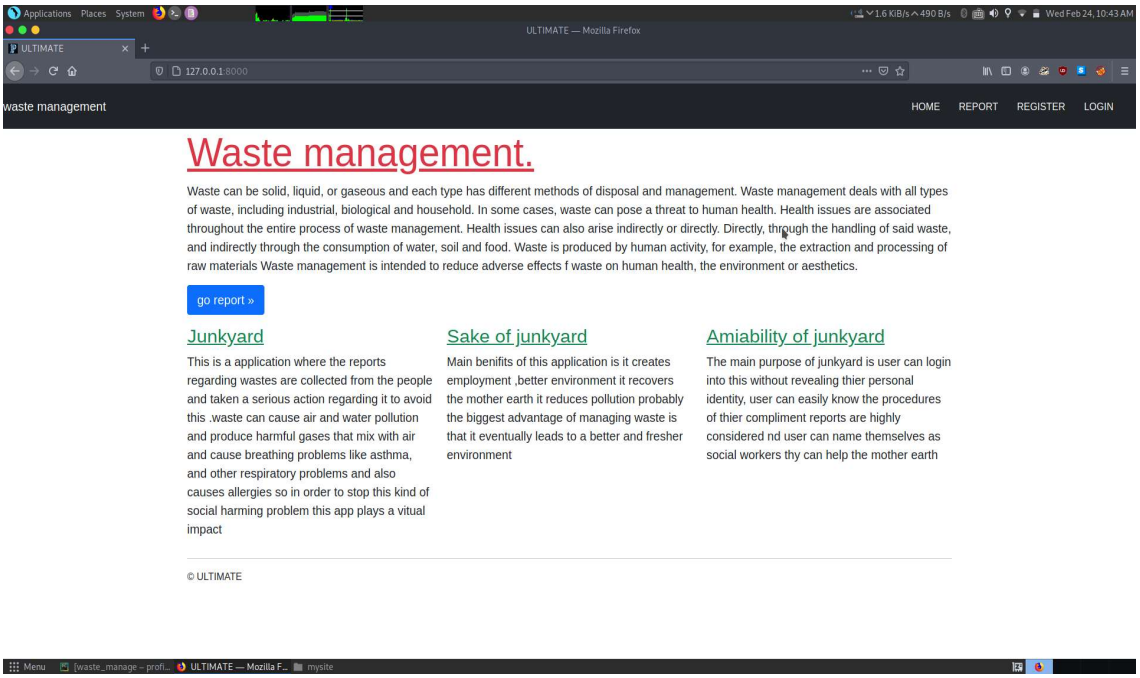

Figure 4.1: Home Page

Description: This is the page where usage of the application is mentioned.

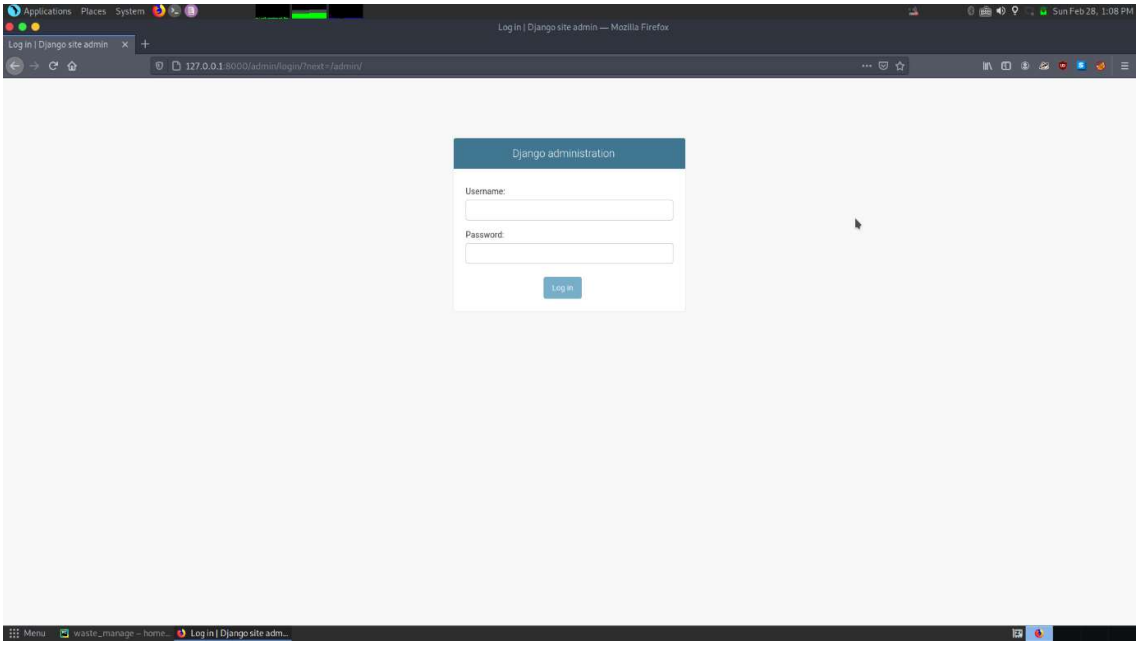

Figure 4.2: Admin Login Page

Description: This is the page where admin can login.

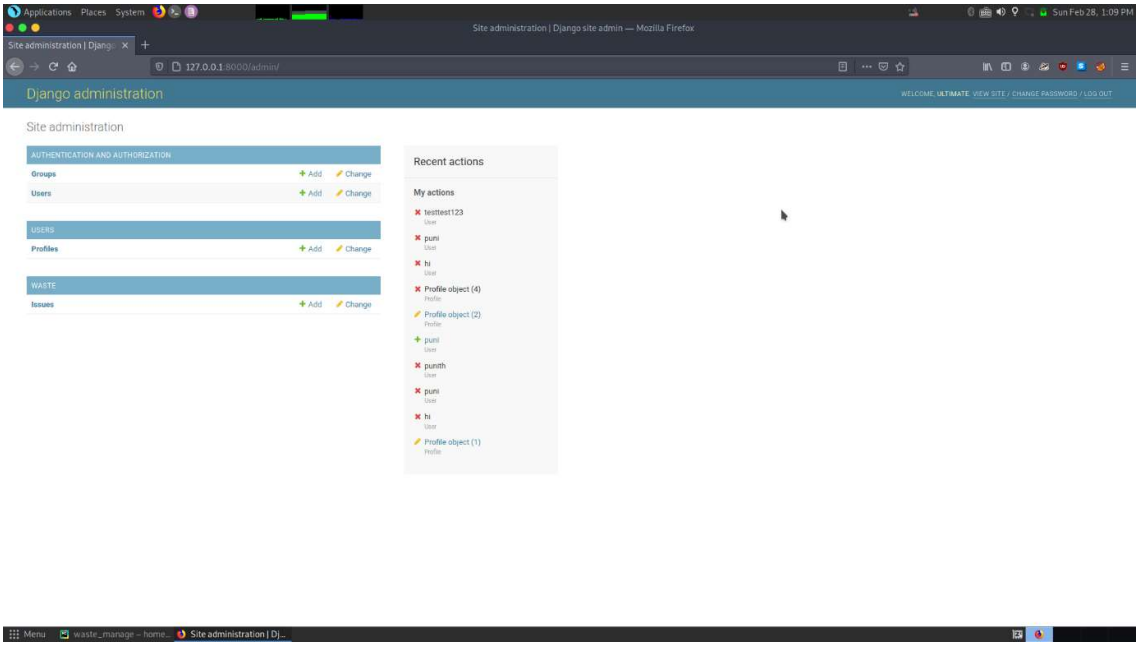

Figure 4.3: Admin Home Page

Description: This is the home page for Admin.

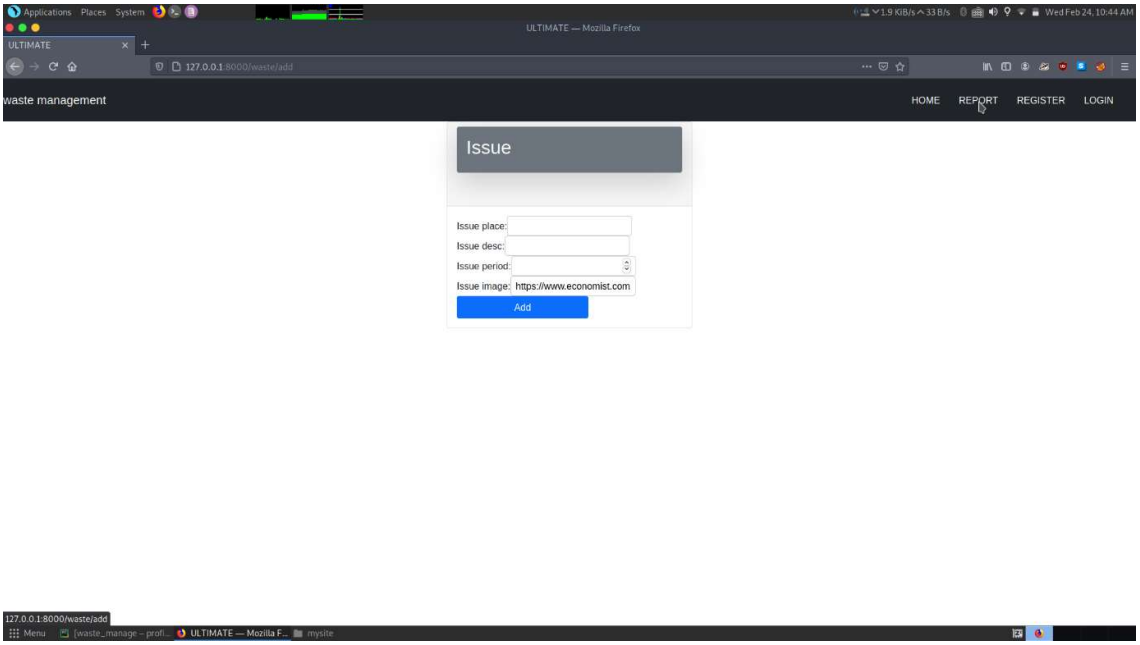

Figure 4.4: Add Issue

Description: In this page where user can enter the issue details.

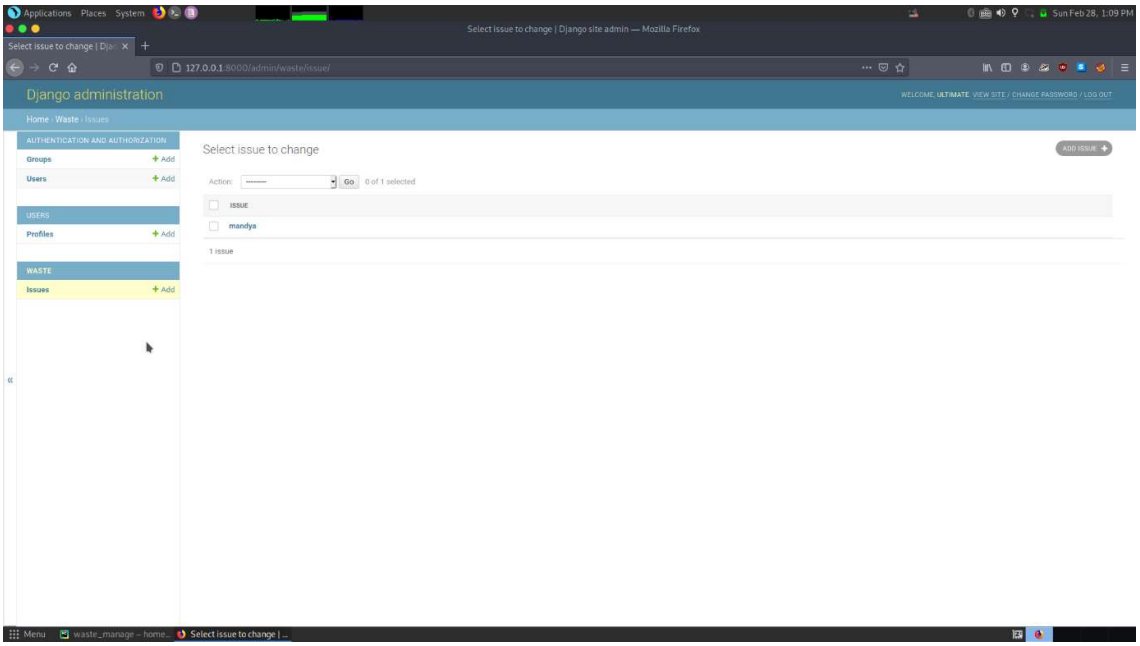

Figure 4.5: Manage Issue

Description: In this page where they can edit their reported issue.

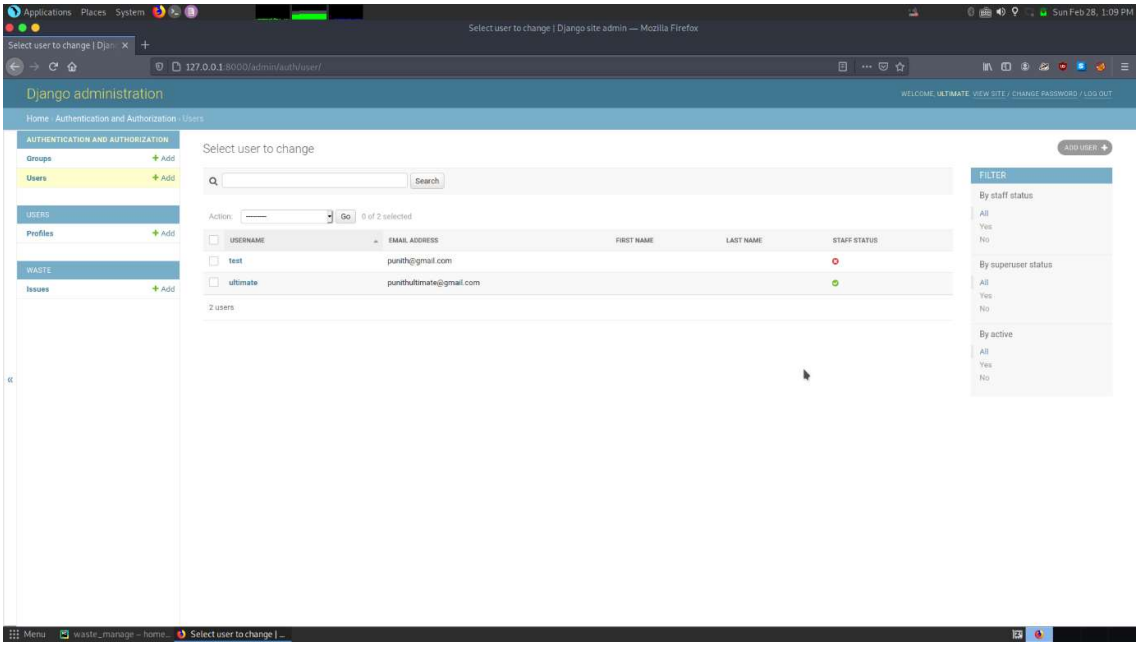

Figure 4.6: Manage User Detail

Description: In this page where Admin can manage the user details

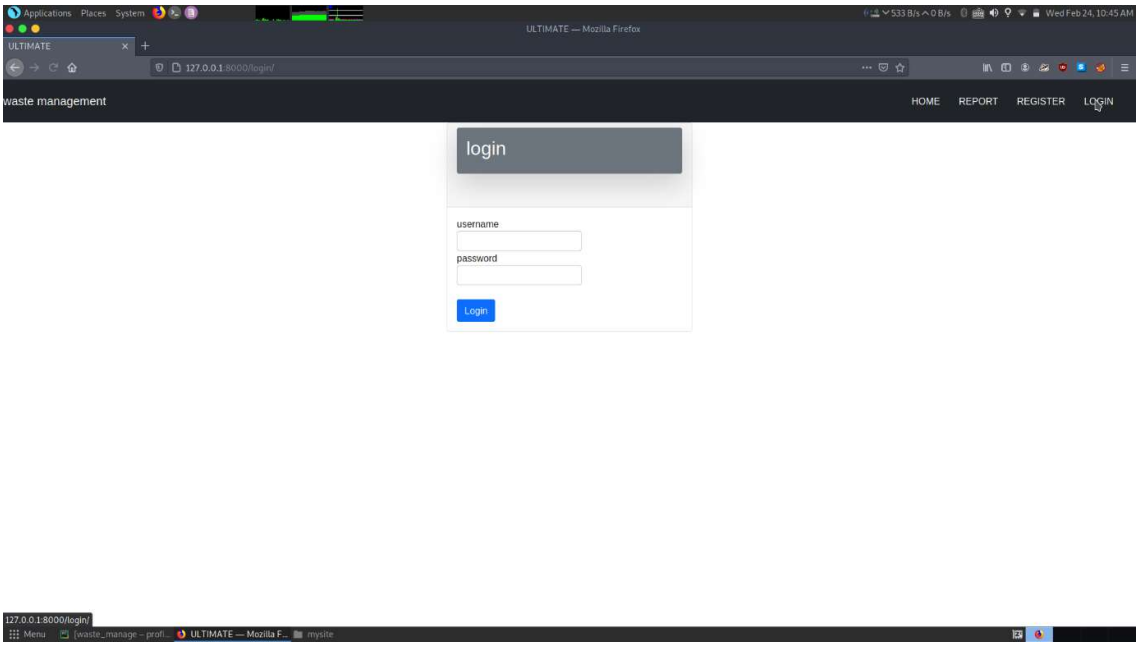

Figure 4.7: User Login Page

Description: In this page where user can login.

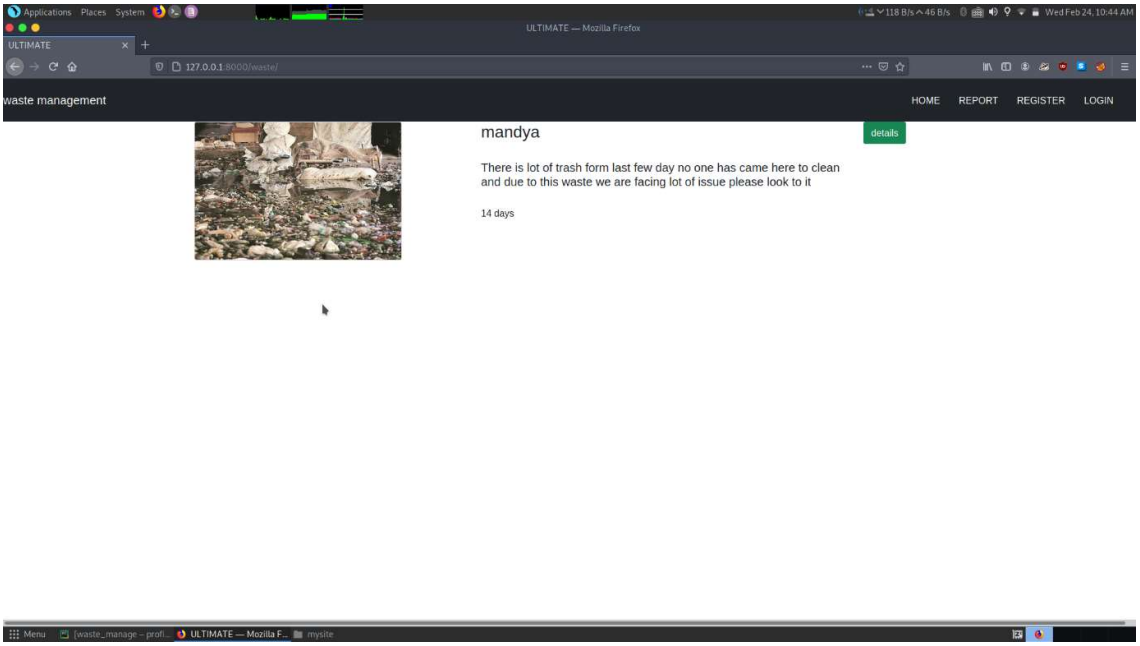

Figure 4.8: Reported Page

Description: In this page where all the reports can be seen.

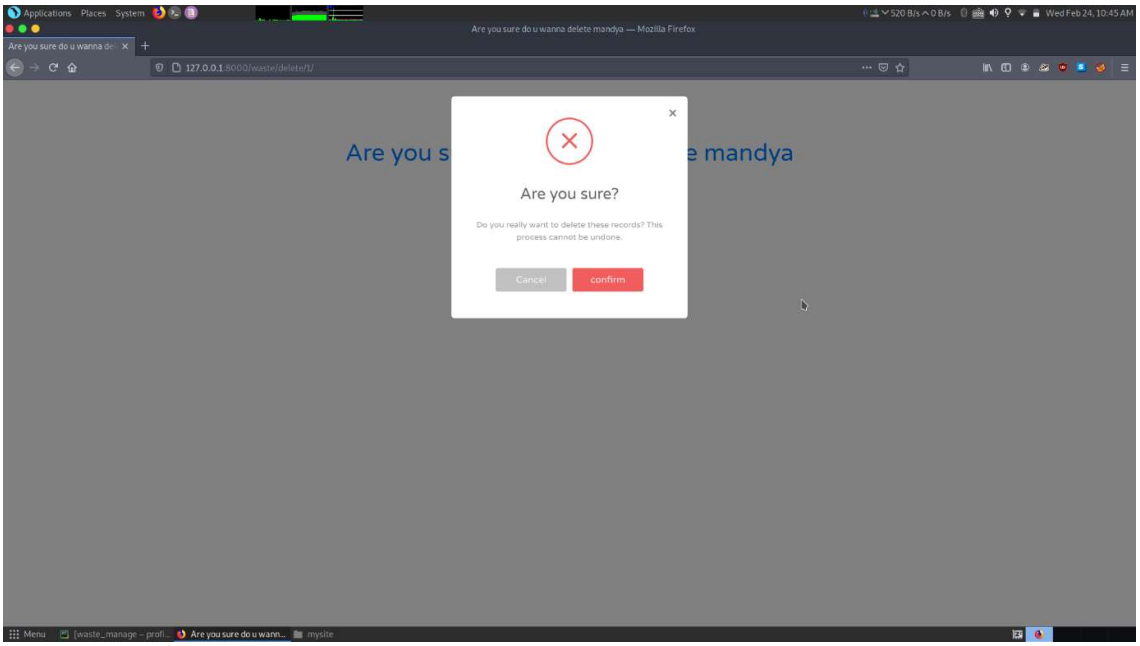

Figure 4.9: Confirm Delete Page

Description: In this page where they can delete the issue.

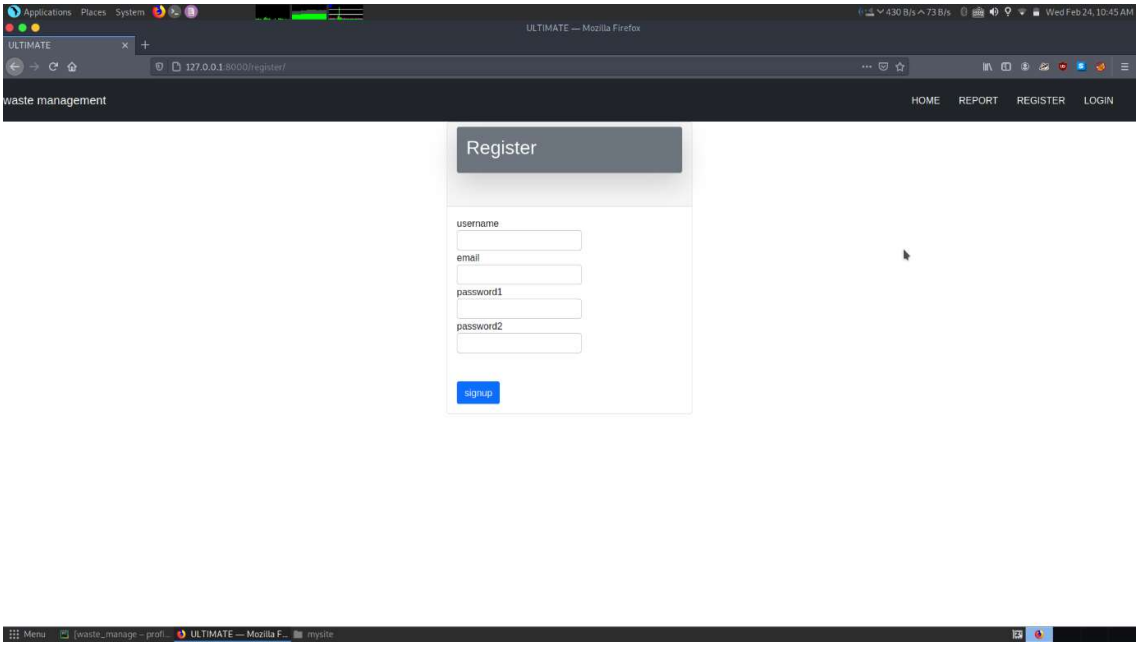

Figure 5.0: User Signup Page

Description: In this page where new user can register.

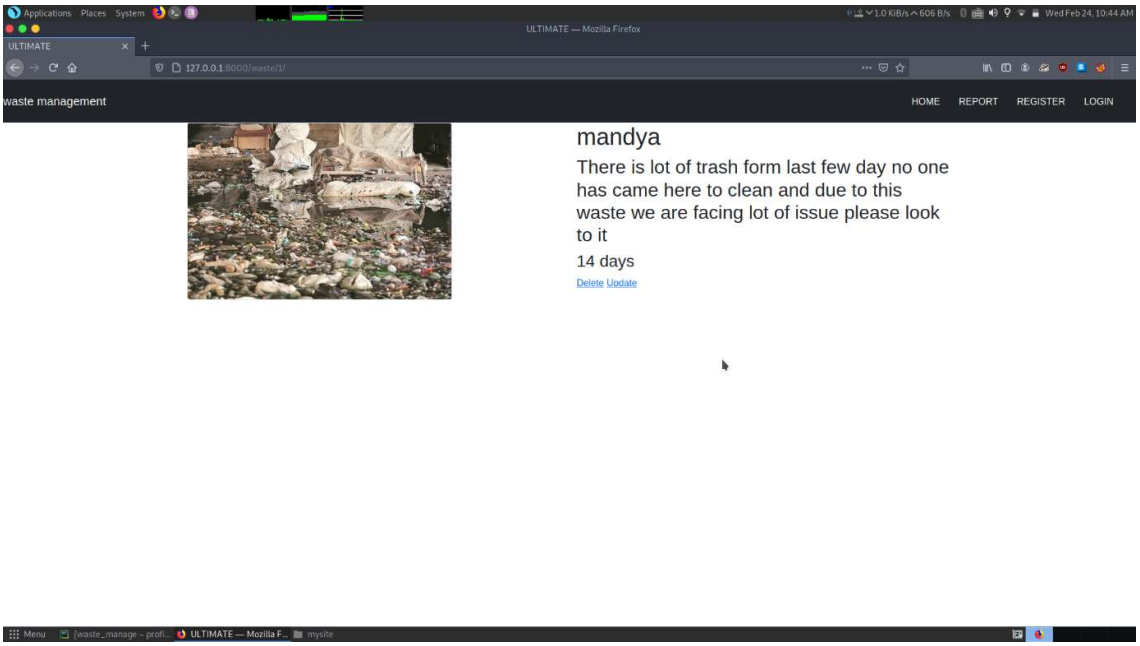

Figure 5.1: Manage Issue for User Page

Description: In this page where Admin can delete the issue.

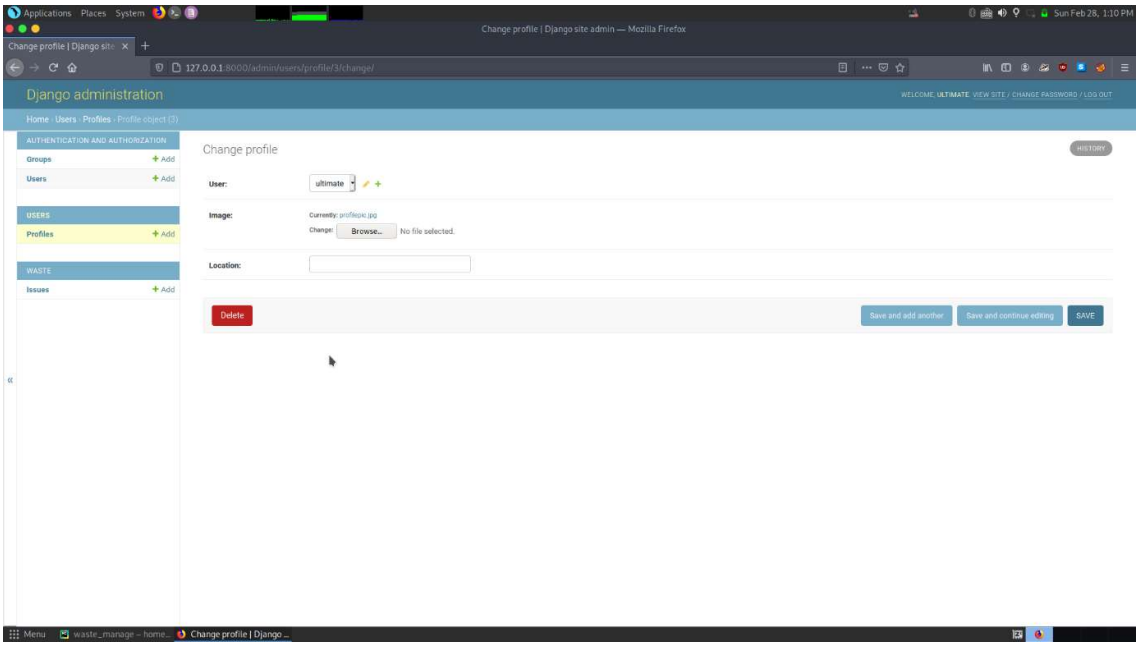

Figure 5.2: Admin Manage Profile

Description: In this page where Admin can update their profile.

CONCLUSION AND FUTURE SCOPE

Conclusion

The developed project fulfills the website facilities estimated for pahse-1 development and as per all the currently addressed requirements of the client (public).

Development team will provide

- Uploading and trail running of the website.
- Plan to avoid/handle unexpected damages
- Probable list of modifications that will guide the pahse-2 development of the project.

Future Enhancement

In future this project can be enhanced to notify the users through mails or messages when there are any updates from the BBMP.

REFERENCES

[1] Database Management System by Prof. Date, A Kannan, S Swamynatham

[2] Database Management System by Prof. S NandaGopalan

[3] www.djangodocumentation.com

[4] www.**python.org/**

[5] www.Bootstrap.com

[6] en.wikipedia.org/wiki/**django**