ADITYA COLLEGE OF ENGINEERING

(Affiliated to JNTUK, Kakinada & Approved by AICTE, New Delhi) Surampalem, A.D.B ROAD, East Godavari District

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project report entitled, **" DISASTER MANAGEMENT ",** **done by M.DURGA PRASAD(16MH1A0596), P.TEJASWINI VENKATA SAI(16MH1A05A5), K.V.V.S.RAVI(16MH1A0587), ACHUER AYUEL(16MH1A05C9), G.HARSHA VARDHAN (16MH1A0580),** at the college for the award of **BACHELOR Of Technology in COMPUTER SCIENCE & ENGINEERING during the academic year of 2016-2020 From Aditya College of Engineering, Surampalem.**

PROJECT GUIDE	**HEAD OF THE DEPARTMENT**
V. ANANTHA LAKSHMI	**Dr. G.S.N.MURTHY M.Tech, Ph.D**
Assoc. Professor	**Professor**
Department Of CSE	**Department Of CSE**

EXTERNAL EXAMINER

A PROJECT REPORT ON

DISASTER MANAGEMENT

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE & ENGINEERING

Submitted By

M. DURGA PRASAD	16MH1A0596
P. TEJASWINI VENKATA SAI	16MH1A05A5
K. V. V. SATYA RAVI	16MH1A0587
ACHUER AYUEL	16MH1A05C9
G. HARSHA VARDHAN	16MH1A0580

Under the esteemed supervision of

V.ANANTHA LAKSHMI

Assoc. Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ADITYA COLLEGE OF ENGINEERING

(Affiliated to JNTUK, Kakinada & Approved by AICTE, New Delhi, Accredited by NAAC)
Surampalem, A.D.B ROAD, East Godavari District

2016-2020

DECLARATION

We hereby declare that this project entitled "**DISASTER MANAGEMENT**" has been undertaken by us and this work has been submitted to ADITYA COLLEGE OF ENGINEERING Affiliated to JNTU Kakinada, in partial fulfillment of the requirements for the award of the Degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING

Project Associates

M.DURGAPRASAD	16MH1A0596
P. TEJASWINI VENKATA SAI	16MH1A05A5
K. V. V. SATYA RAVI	16MH1A0587
ACHUER AYUEL	16MH1A05C9
G.HARSHA VARDHAN	16MH1A0580

ACKNOWLEDGMENT

It is with immense pleasure that we would like to express our indebted gratitude to our guide **V. Anantha Lakshmi** who has guided us a lot and encouraged us in every step of project work, her valuable moral support and guidance throughout the project helped us greater extent.

We are ever grateful to **Dr. G.S.N. Murthy,** Head of the Department of CSE, for his valuable guidance given to us throughout the period of our project.

Our sincere appreciation to Principal, of Aditya College Of Engineering, **DR. A. Ramesh** for his cooperation and helping in completion of our project and throughout our course.

We avail this opportunity to express our deep sense and heart full thanks to the **Management of Aditya College of Engineering** for providing a great support for us in completing our project by arranging the facilities and the resources needed to complete our project and for giving us the opportunity of doing the project.

We would like to thank our **Parents** for inspiring us all the way and for arranging all the facilities and resources needed for our project. Not to forget, our **Teaching, Non-Teaching staff** and our friends who has directly or indirectly helped and supported us in completing our work successfully within the given time.

Project Associates

M.DURGAPRASAD	16MH1A0596
P. TEJASWINI VENKATA SAI	16MH1A05A5
K. V. V. SATYA RAVI	16MH1A0587
ACHUER AYUEL	16MH1A05C9
G.HARSHA VARDHAN	16MH1A0580

ABSTRACT

A disaster is a serious disruption in the functioning of a community or a society, which exceed the ability of the affected community or the society to cope using its own resources; for instance, there is a widespread human, economic, environmental and material impact due to turbulent cyclones. **Disaster Management** refers to conservation of lives and property during a natural and man-made disaster. Disaster management plans are multi-layered and are planned to address issues such as floods, hurricanes, fires, mass failure of utilities, rapid spread of disease and droughts. In this disaster

With the help of this project we can prevent the damage occurred to some extent. This project management we have three phases with respect to the stages of disaster is made with the Moto to help people. We use **Django** in this project. **Django** is a high-level Python web framework that encourages rapid development and clean pragmatic design. A Web framework is a set of components that provide a standard way to develop websites fast and easily. Django's primary goal is to ease the creation of complex database-driven websites. In Django, we have built-in database to use. We maintain the data that is required during disaster like safe places in that area, transport facilities to safe place and medical requirements. Here safe place means the place which is not affected due to disaster and it should contain minimum facilities.

INDEX

Contents		Page No.
1. INTRODUCTION		**1**
2. REQUIREMENT		**2**
2.1	Hardware & software requirements	3
2.1.1	Hardware requirements	3
2.1.2	Software requirements	3
2.2	Software requirements specification	4
2.2.1	Vision	4
2.2.2	Scope	4
2.2.3	Glossary	4
2.2.4	System function	4
2.2.5	Detailed software requirement	5
2.2.6	List of use cases	6
2.2.7	Brief use case description	6
2.2.8	Function capabilities	7
2.2.9	Security requirements	8
2.2.10	Non functional requirements	8
3. SYSTEM ANALYSIS		**9**
3.1	Existing system	9
3.1.1	Limitations of existing system	9
3.2	Proposed system	9
3.2.1	Advantages of proposed system	9
3.2.2	Project Module	10
4. SOFTWARE ENVIRONMENT		**11**
4.1	Selected Software	11
4.1.1	Introduction to Django	11
4.2	System architecture	13
5. SYSTEM DESIGN		**14**
5.1	UML DIAGRAMS	15
5.1.1	Use case diagrams	15
5.2	Class diagrams	18
5.3	Sequence diagram	20
5.4	Activity diagram	21
6. SAMPLE CODE		**23**
6.1	views.py	26

6.2 Urls.py(disaster) 29
6.3 Urls.py(webpage) 29
6.4 Manage.py 30
6.5 Settimgs.py 30
6.6 Apps.py 33
6.7 Asgi.py 33
6.8 Wsgi.py 33
6.9 Models.py 34
6.10 admin.py 34
6.11 HTML Files 35

7. TEST CASES 44
7.1 Introduction 44
7.2 Types of testing 44
7.2.1 Unit testing 44
7.2.2 Integration testing 44
7.2.3 Functional testing 44
7.2.4 System testing 45
7.3 Test cases 45

8. OUTPUT SCREENS 48

9. CONCLUSION AND FUTURE SCOPE 52

10. BIBILOGRAPHY 53
10.1 Reference ad website references 53

LIST OF FIGURES

Fig. no	Name of the Figure	Page. No
4.2	System architecture	13
5.1.2	Use case diagram for admin	16
5.1.3	Use case diagram for Government Official	16
5.1.4	Use case diagram for User	17
5.2.2	Class diagram for Disaster Management	19
5.3	Sequence diagram for Disaster Management	20
5.4.1	Activity diagram for Admin	21
5.4.2	Activity diagram for Government officials	22
5.4.3	Activity diagram for User	22
8.1	Index page	48
8.2	Safe places	49
8.3	Admin login	49
8.4	Administration site	50
8.5	Government official login	50
8.6	Forgot password	51
8.7	Adding new data	51

1. INTRODUCTION

The term “Disaster Management” encompasses the complete realm of disaster-related activities. Traditionally people tend to think of disaster management only in terms of the post-disaster actions taken by relief and reconstruction officials; yet disaster management covers a much broader scope, and many modern disaster managers may find themselves far more involved in pre-disaster activities than in post-disaster response.

Disaster management essentially deals with management of resources and information towards a disastrous event and is measured by how efficiently, effectively and seamlessly one coordinates these resources. The ability to effectively deal with disasters has become a challenge to modern technology. We designed the website which helps to the people to survive. In this website we will have information about safe places and other information regarding it. User can view these details by entering his location.

2. REQUIREMENT ANALYSIS

When the organization or a project development environment takes up a project, the first and foremost analysis performed is the requirement analysis. A "problem definition" is formulated. This is a very eminent phase in the project development. Utmost care has to be taken in this stage of project development if the analysis goes wrong and the problem definition quotes a wrong requirement specification the project developed will not be of use.

Requirement analysis is a software engineering task that bridges the gap between system software allocation and software design. It provides the system engineer to specify software functions and performance indicate software's interface with other system elements and establish constraints that software must need. The basic is to obtain a clear picture of the needs and requirements of the end user and also the organization.

The requirements phase basically consists of three activities:

1. Requirement Analysis
2. Requirement specification
3. Requirement validation

REQUIREMENT ANALYSIS:

The basic aim of this stage is to obtain a clear picture of the needs and requirements of the end-user and also the organization. Analysis involves interaction between analyst and the data set.

The essential final consistent specifications are divided into five parts

a. Problem recognition
b. Evaluating And Synthesis
c. Modeling
d. Specification
e. Review

2.1Hardware & software requirements:

2.1.1 Hardware requirements

RAM-32GB OR 16GB

PROCESSOR- INTEL

CORE i5 **SPEED-** 2.8GHz

STORAGE- 300GB

2.1.2 Software requirements:

OPERATING SYSTEM: WINDOWS 10 PRO

TECHNOLOGY: Django

LANGUAGE: PYTHON, HTML, CSS

PYTHON LIBRARIES: Render, http response

2.2 Software Requirement Specifications:

2.2.1 Vision:

For government or an organization which works to reduce or prevent damage caused during floods, the disaster management is a website which can help them to do so.

For user who does not know much about the nearby safe places, hospitals and transport details, the disaster management is a website which can help them to find those required places.

2.2.2 Scope:

This project is aimed at developing a website that is used for government to reduce the damage caused by floods. The main use of this application is to help the users to save their life during floods.

In this system, government officials will add safe places information, rescue team information, transport system details. As it was a online application

2.2.3 Glossary:

Admin	Register government officials
Government official	Add safe places, give alerts
User	Enter location, view safe places

2.2.4 System Functions:

S.NO	**Description**
	ADMINISTRATOR
1	Login into the system
2	Add government officials
3	Delete government officials
4	Logout
	GOVERNMENT OFFICIALS
1	Login
2	Add safe places information
3	Give alerts
4	Logout
	USER
1	Visit webpage
2	Enter location
3	View safe places information

2.2.5 Detailed Software Requirements:

2.2.5.1 Actors:

Actor Name	Government Official
Description	Add safe place details and give alerts.
Main Activities	They can add safe place details like it’s name, distance, contact number to the system and can give alerts during floods
Frequency of Use	High
Work Environment Location	Browser
Number of Users	Any number

Actor Name	Administrator
Description	Handles all admin related tasks throughout the application.
Main Activities	Authenticate government officials & Uses the system to setup initial data, define access control etc.
Frequency of Use	Medium
Work Environment location	Browser
Number of Users	Any number

Actor Name	User
Description	Enter location and view safe place details
Main Activities	They will enter their current location and then they will view the details of safe places mentioned in the website during floods.
Frequency of Use	High
Work Environment Location	Browser
Number of Users	Any number

2.2.6 List of Use Cases:

- Get Registered
- Login
- Add Government Officials
- Delete Government Officials
- Add Safe Place Details
- Give Alerts
- Enter Location
- View Safe Place Details

2.2.7 Brief Use Case Description:

Use Case name	**Login**
Actors	Admin, Government Official
Goal	Login in to the system
Description	Admin & Government Official login into the system
Precondition	None
Post condition	Admin & Government Official can access the system

Use Case name	**Add Safe Place details**
Actors	Government Official
Goal	Add new safe place details
Description	Government Official add safe place details into the system
Precondition	None
Post condition	None

Use Case name	**Give Alerts**
Actors	Government Official
Goal	Give alerts in the system
Description	Government Official give alert in the system about flood
Precondition	None
Post condition	None

Use Case name	**View Safe Place details**
Actors	User
Goal	View safe place details
Description	User can view safe place details in the system
Precondition	None
Post condition	None

2.2.8 Functional Capabilities:

- Government Officials identities are anonymous.
- Password should be encrypted.

2.2.9 Security Requirements:

2.2.9.1 Administrator Management and Authentication:

2.2.9.1.1 The system must be accessed by an authentic person, i.e. Administrator.
2.2.9.1.2 The adding / deleting of government officials are done by Administrator.

2.2.10 Non-Functional Requirements:

The application should support interruptions for regular network device tasks
i.e. viewing safe place details. (Supportability)

The webpage should load within seconds. (Performance)

Flexible architecture will be highly desirable for the further extension. (Supportability)

The system should be available 24 x 7. (Reliability/ availability)

- **Reliability :**
  The system has been developed by considering every single quality factor.
- **Security :**

  The system has been authorized so that valid user can add / delete details of safe places.
- **Availability :**
  The system has been developed in such a way that loading of safe place details must be finished within span of time.

3. SYSTEM ANALYSIS

3.1 Existing System

- The current system is maintaining manually.
- Manual maintenance of records involves burden and it is quite tedious task. In general existing there is no security.
- If any record missed it is very difficult to retrieve the data.

3.1.1 Limitations of Existing System

- The study was conducted during floods so some sort of disturbances were inevitable.
- Since there is no well-established Disaster Management department in the valley so we couldn’t get the reliable and organized data.
- Primary information was collected directly from flood victims so what they conveyed may be biased.
- There are no alerts in the system about floods.
- It is not available to all people who need it.
- It requires paper work, large amount of laborious work

3.2 Proposed System

- The proposed system is computerized to provide greater easiness and flexible.
- The system is constructed in an object oriented trend thinking in an abstract way considering all the involvements as object.
- With the help of our project, information about safe places and other important contacts will be known to all users.

3.2.1 Advantages of Proposed System

- It is available 24 x 7.
- It can give alerts / warnings to the users before occurrence of floods.
- The primary information is collected by government officials so that no fake information is present in system.

- As it is a web based project it can be used by all in their time of need.
- Adding of new safe place details is easier than existing system.

3.2.2 Project Module:

1. **Admin Module:**

   The admin is responsible for maintenance of system. He can add or delete the government officials to the system.

2. **Government Officials Module:**

   Government official must be an authorized user. He can add safe place details, transport details and rescue team details. He can also give alerts before occurrence of floods.

3. **User Module:**

   User does not have any authorization. If user has authorization, only authorized users will know about the safe places and the unauthorized users do not know about it due to this their life may be in danger. So there is no authorization for user. User will enter his location and then he can view safe place details.

4. SOFTWARE ENVIRONMENT

4.1 Selected Software:

4.1.1 Introduction to Django:

Django is a Python-based web framework which allows you to quickly create web application without all of the installation or dependency problems that you normally will find with other frameworks. When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django gives you ready-made components to use.

Features:

- Don't repeat yourself scenario
- Unbelievably fast
- The complete Package
- Tons of packages
- Highly secure
- Widely Scalable
- Incredibly Versatile
- Multi-lingual support

Installation of Django:

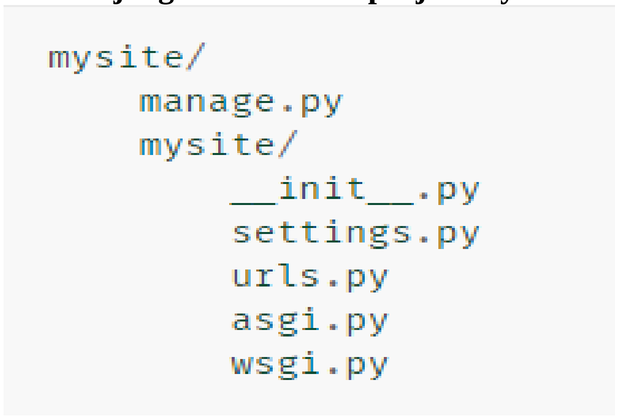
Install python3 if not installed in your system (according to configuration of your system and OS). Try to download the latest version of python is python3.6.4 this time.

Command to install Django

- **Pip install Django**

Command to start project

- **django -admin startproject mysite**



```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```

These files are:

The outer **mysite/** root directory is a container for your project. Its name doesn't matter to Django; you can rename it to anything you like.

manage.py: A command-line utility lets you interact with this Django project in various ways. You can read all the details about **manage.py** in django-admin and manage.py.

mysite/_init_.py: An empty file that tells Python that this directory should be considered a Python package. If you're a Python beginner, read more about packages in the official Python docs.

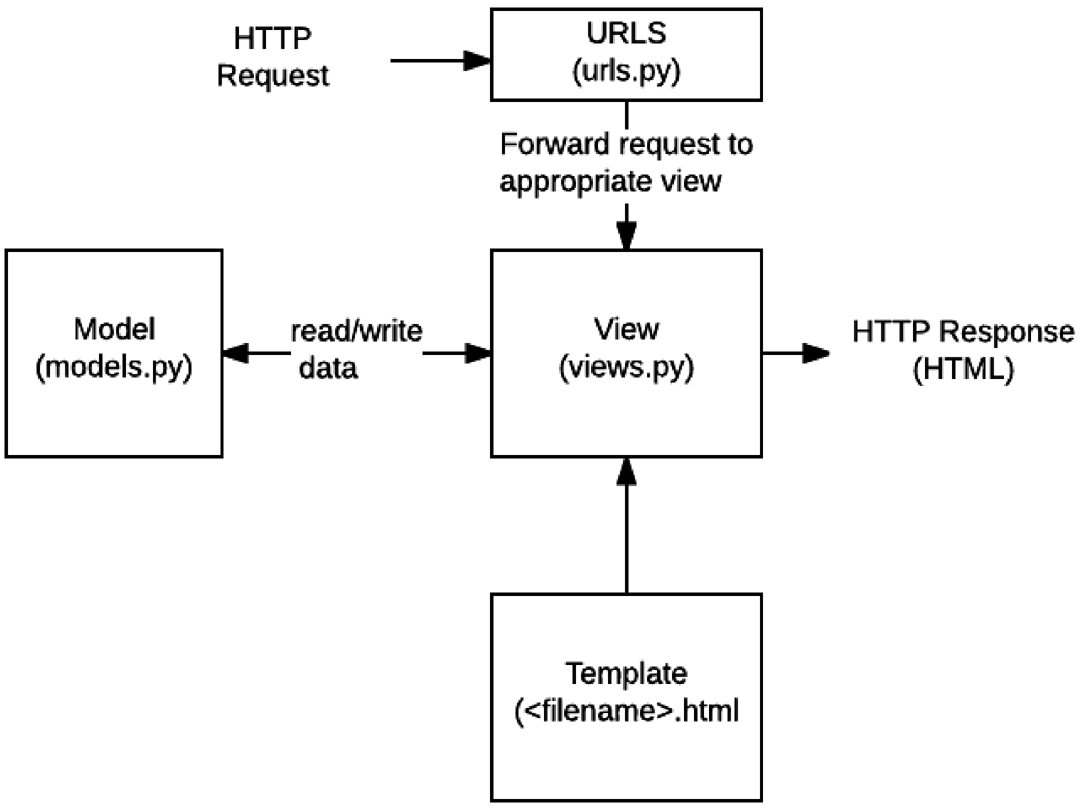
mysite/ settings.py: Settings/configuration for this Django project. Django settings will tell you all about how settings work.

mysite/ urls.py: The URL declarations for this Django project; a "table of contents" of your Django-powered site. You can read more about URLs in URL dispatcher.

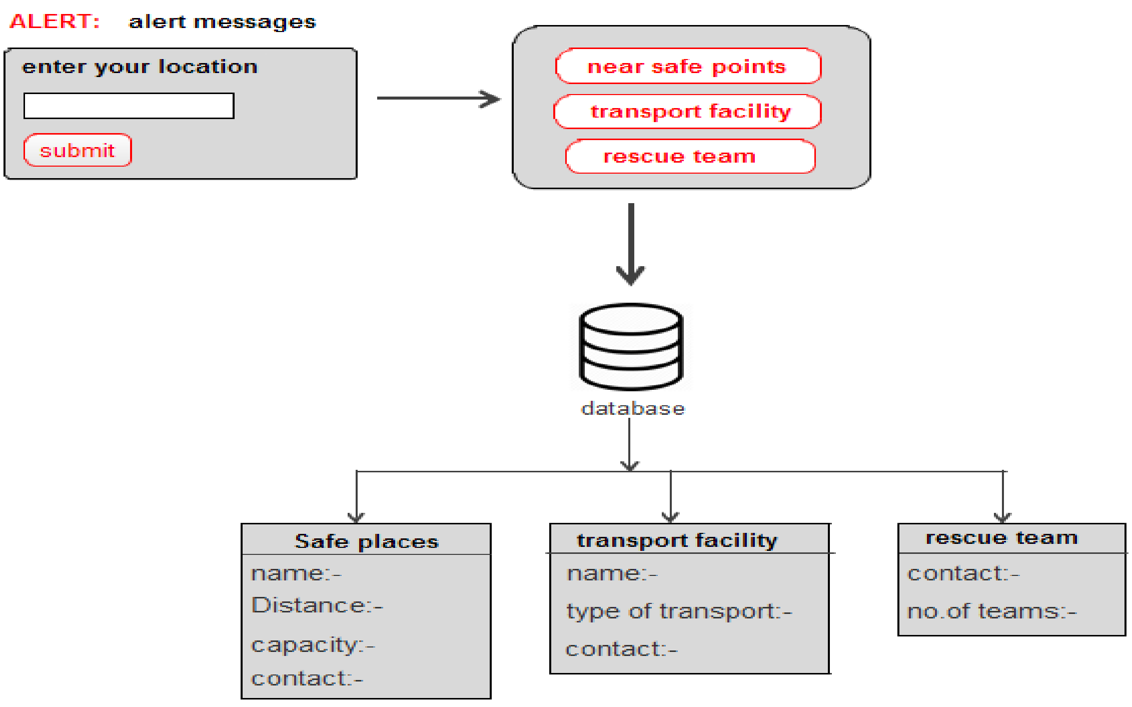
mysite/ asgi.py: An entry-point for ASGI-compatible web servers to serve your project. See How to deploy with ASGI for more details.

mysite/ wsgi.py: An entry-point for WSGI-compatible web servers to serve your project. See how to deploy with WSGI for more details.

Working:



4.2 System Architecture:



5. SYSTEM DESIGN

In this design phase we design the system making use of study phase and the data flow diagrams. We make use the general access methods for designing. We consider the top down approach. In the design phase we determine the entities and their attributes and the relationships between the entities. We do both logical and the physical design of the system.

The most creative and challenging phase of the life cycle is system design. The term design describes a final system and the process by which it is developed. It refers to technical specifications that will be applied in implementations of the candidate system. The design may be defined as “the process or a system with sufficient details to permit its physical realization.

The designer goal is how the output is to be produced and in what format. Samples of output and input are also presented. Second input files and database files have to be designed to meet the requirements of the proposed output. The processing phases are handled through the program construction and testing. Finally, details related to justification of the system and an estimate of the impact of the candidate system on the user and the organization are documented and evaluated by management as a step towards implementation.

The importance of software design can be stated in single word “Quality”. Design provides us with representations of software that can be assessed for quality. Design is the only way where we can accurately translate a costumer’s requirement into complete software product or system. Without design we risk building a unstable system that might fail if small changes are made. So it is an essential phase in the development of software product.

System design develops the architectural detail required to build a system or product. The system design process encompasses the following activities.

- Partition the analysis model into subsystems.
- Identify the concurrency that is dictated by the problem.
- Allocate subsystems to processors and tasks.
- Develop a design for the user interface.
- Choose the basic strategy for implementing data management.
- Identify global resources and control mechanisms required to access them.

5.1 UML diagrams

5.1.1 Use case diagram:

The unified modeling language allows the software engineers to express an analysis model using the model notation that is governed by asset of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly perspective. Each view is defined by diagrams, which is as follows.

User model view:

This View Represents the System from the User's Perspective. The Analysis Is Representation Describes A Usage Scenario From The End-User Perspective.

Structural model view:

In this model the data and functionality are arrived from inside the system. This model view models the static structures.

Behavioral view:

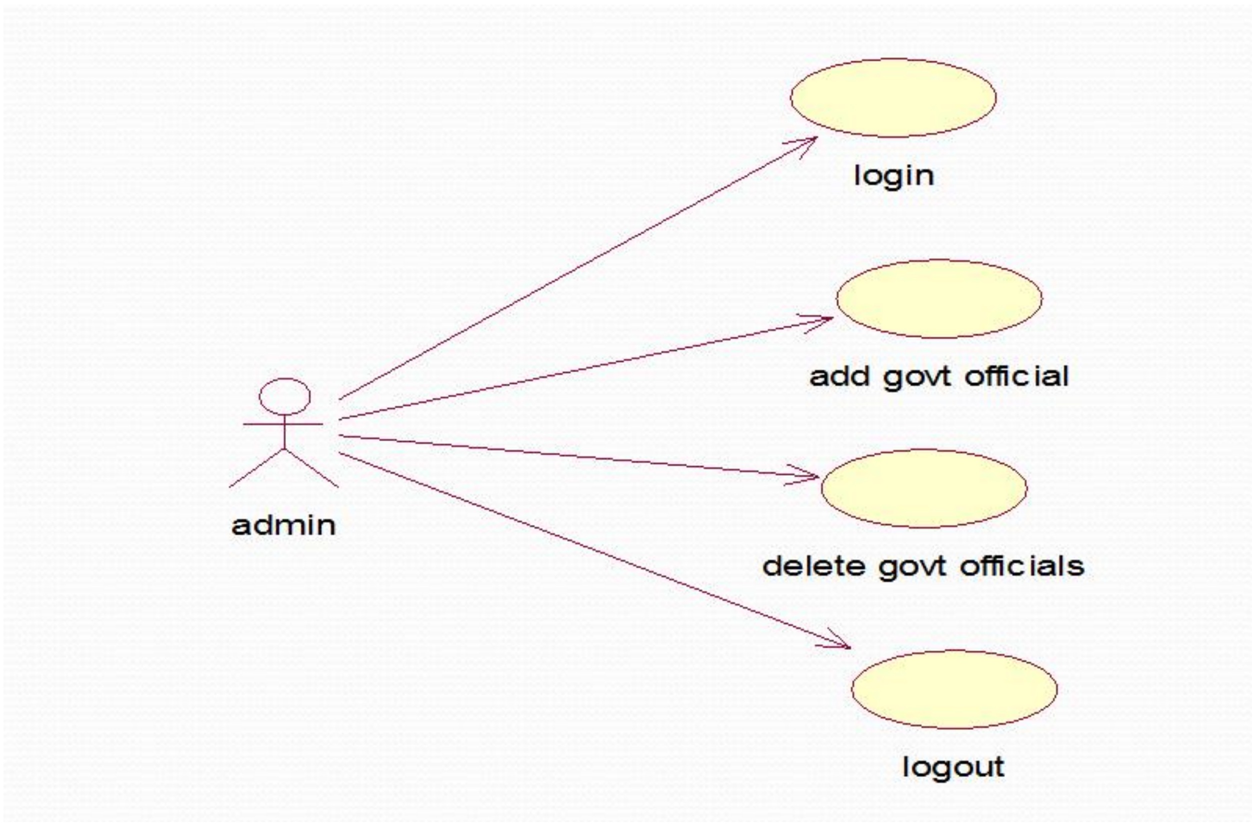
It represents the dynamic of behavioral as parts of the system, depicting the interactions of collections between various structural elements described in the user model and structural model view.

Implementation model view:

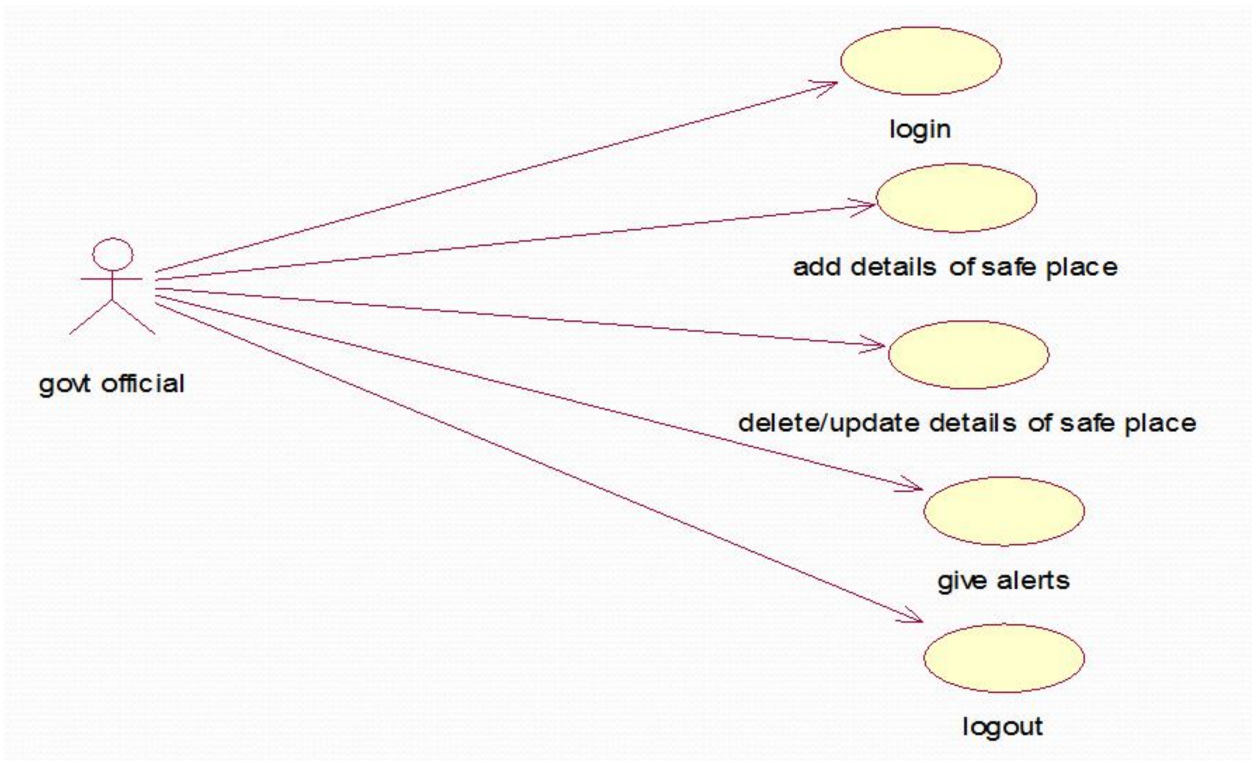
In this the structural and behavioral as parts of the system are represented as they are to be built.

Use case diagrams are created to visualize the relationships between actors and use cases. A use case is a pattern of behavior the system exhibits. Each use case is a sequence of related transactions performed by an actor and the system. Diagrammatically actor and use case are represented by stick figure and oval respectively.

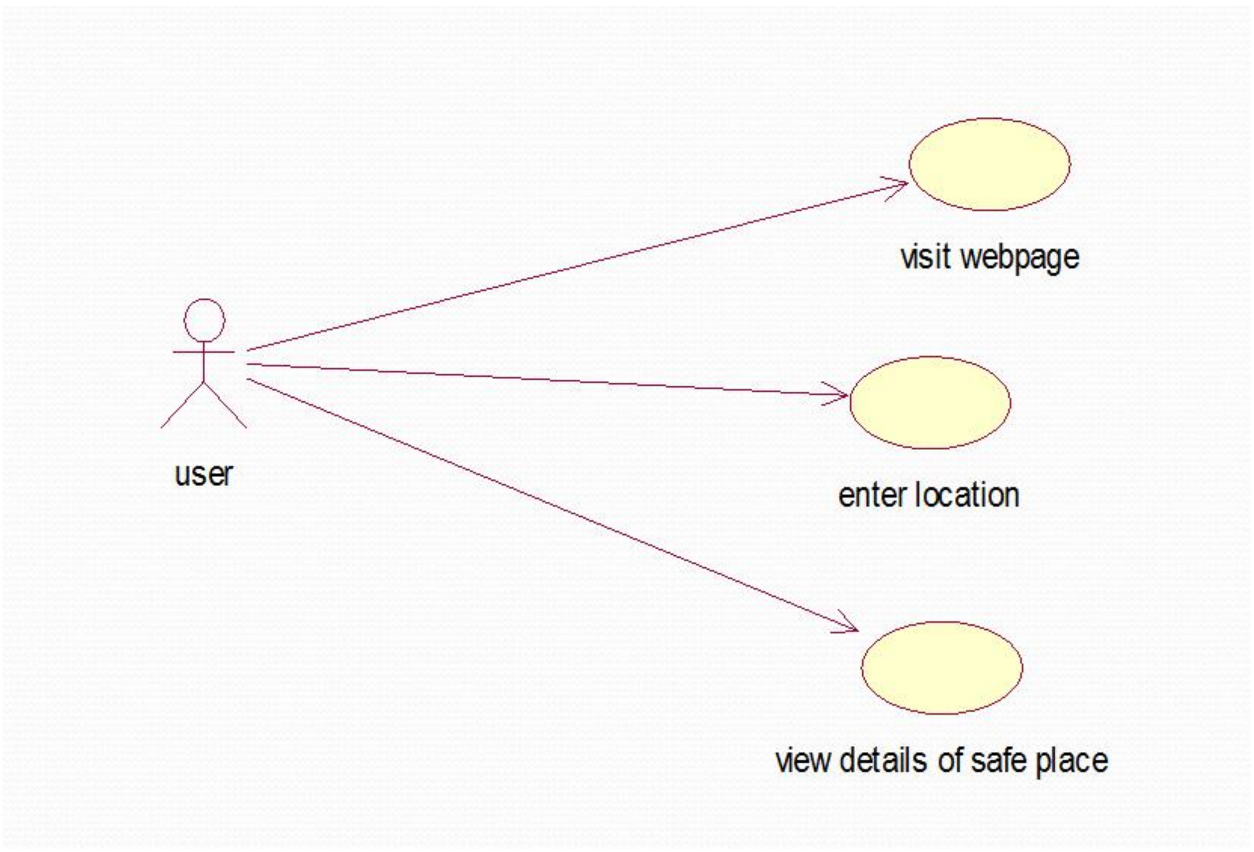
5.1.2 Use case diagram for admin:



5.1.3 Use case diagram for government official:


login
add details of safe place
govt official
delete/update details of safe place
give alerts
logout

5.1.4 Use Case diagram for user:



5.2 Class diagram:

A class diagram show asset of class interfaces and collaborations and their relationships these diagrams are most common diagram found in modeling object oriented system class diagram address the static design view of system.

Relationships there are four kinds of relationships in UML they are

1. Dependency
2. Association
3. Generalization
4. Realization

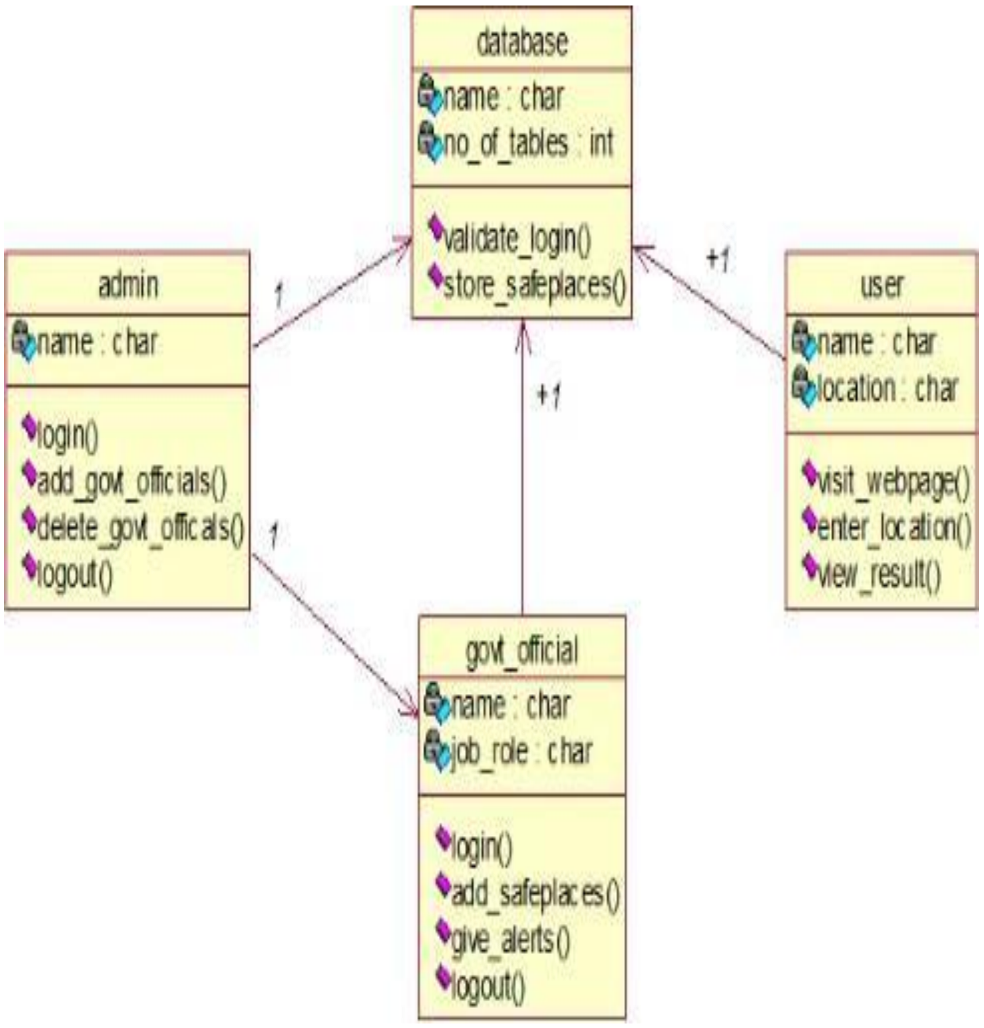
1. Dependency:
It is a semantic relationship between two things in which a change one thing may affect the semantics of the other things graphically it is a represented line possibility directed and occasionally including a label.

2. Association:
Association is a structural relationship that describes a set of links being a collection among objects graphically association is represented as solid line.

3. Generalization:
It is a special kind of association representing a structural relationship between a whole and its part graphically it is represented as solid line with diamond.

4. Realization:
It is a semantic relationship between classifiers where in one classifier specifies a contract that another classifier guarantees to carry out. Graphically it is represented as cross between a generalization and a dependency relationship.

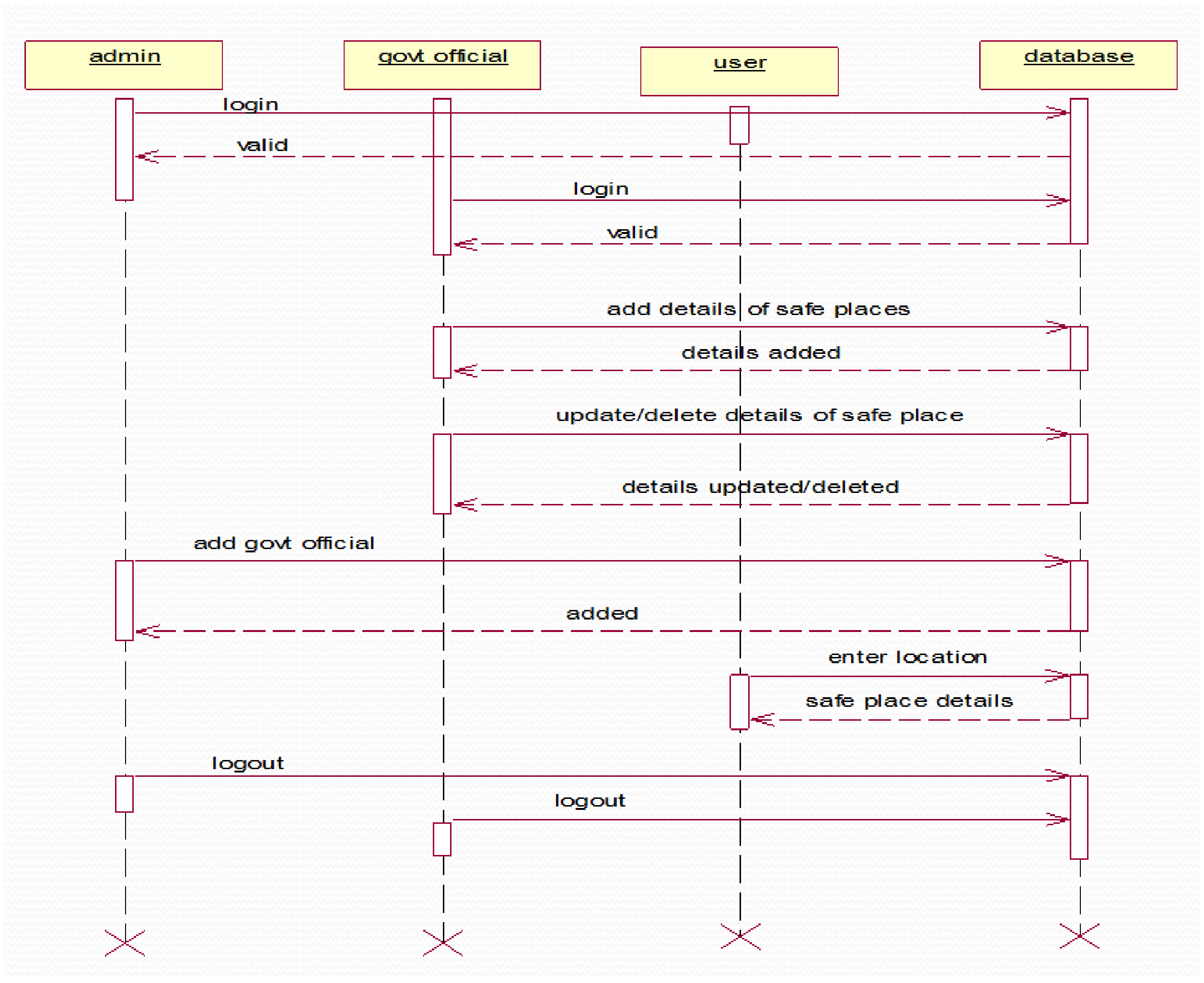
5.2.2 Class Diagram for Disaster Management:



5.3Sequence Diagram:

Sequence diagram is an interaction diagram which focuses on the time ordering of messages. It shows a set of objects and messages exchange between these objects. This diagram illustrates the dynamic view of a system.

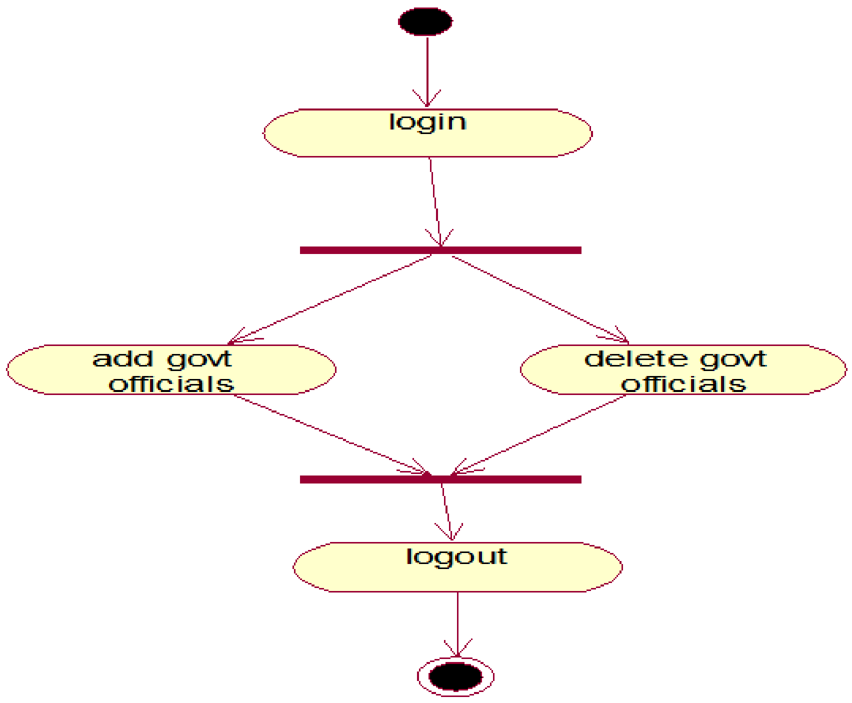
5.3 Sequence Diagram for Disaster Management:



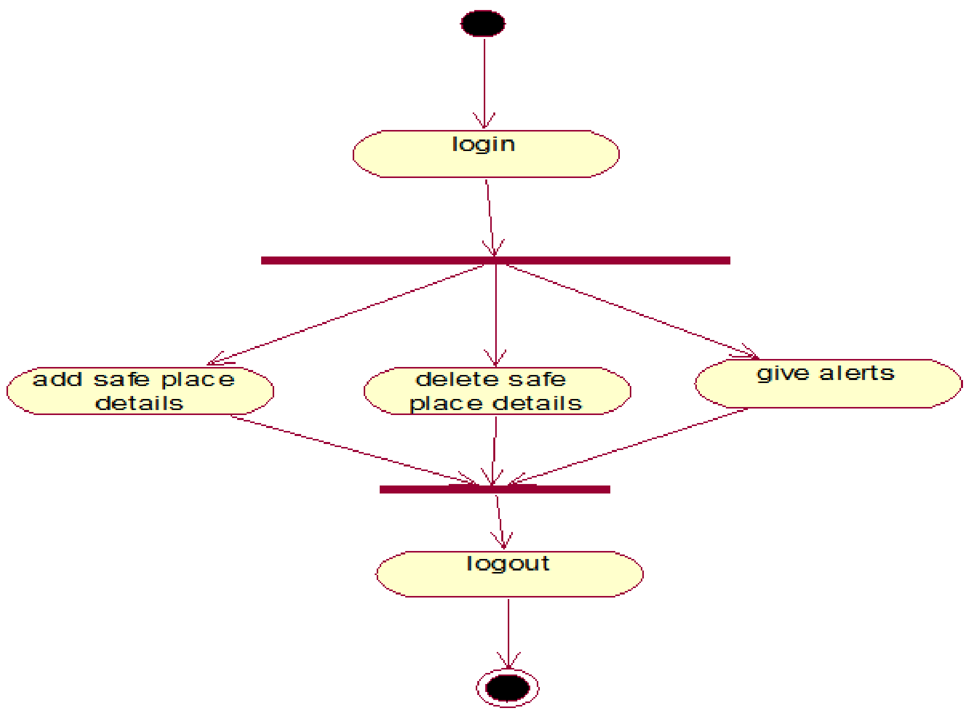
5.4 Activity Diagram:

Activity diagram is basically a flow chart to response the flow form one activity to another activity. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagram deals with all types of flow control by using different elements like Fork, Join.

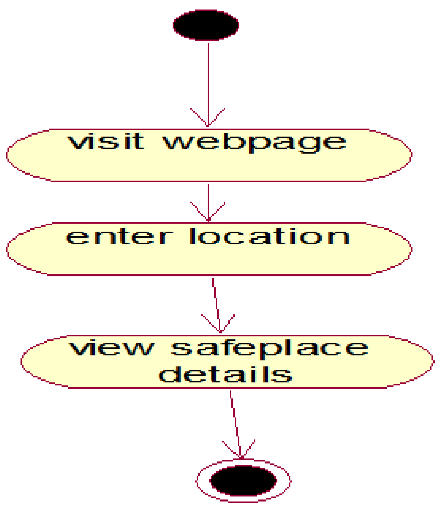
5.4.1 Activity Diagram for Admin:



5.4.2 Activity Diagram for Government Official:



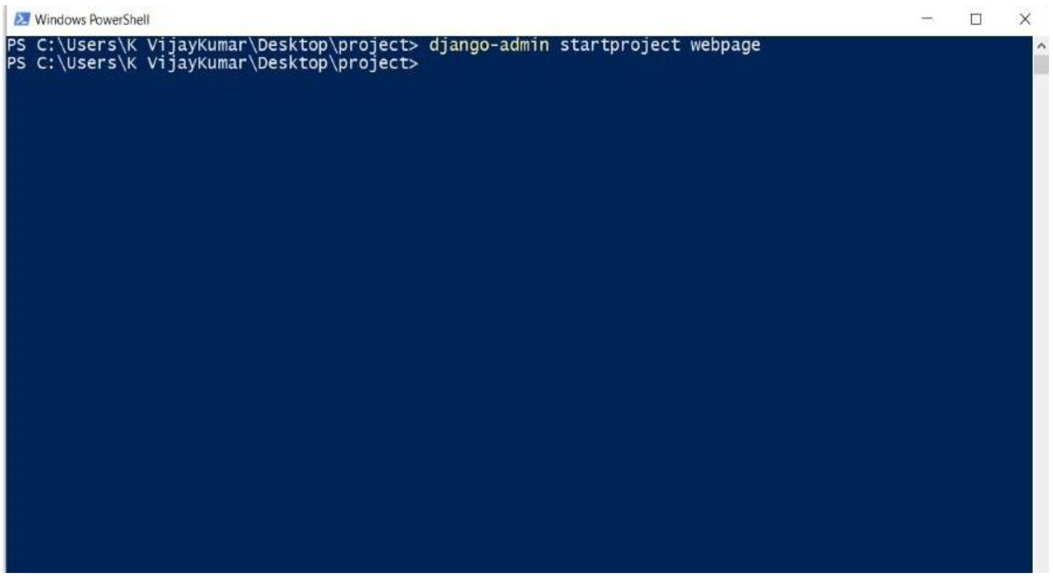
5.4.3 Activity Diagram for User:



6. SAMPLE CODE

First we have to enter the following command to initialize a project

- ➢ django-admin startproject webpage

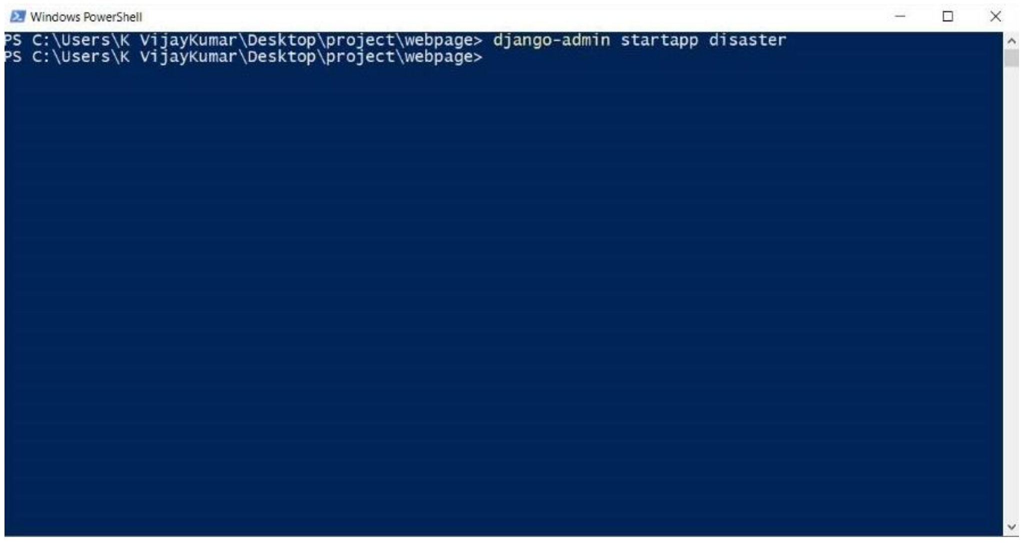


Windows PowerShell

```
PS C:\Users\K VijayKumar\Desktop\project> django-admin startproject webpage
PS C:\Users\K VijayKumar\Desktop\project>
```

We know that a project is a collection of apps. In order to create an app named disaster, we need to execute the following command

- ➢ django-admin startapp disaster

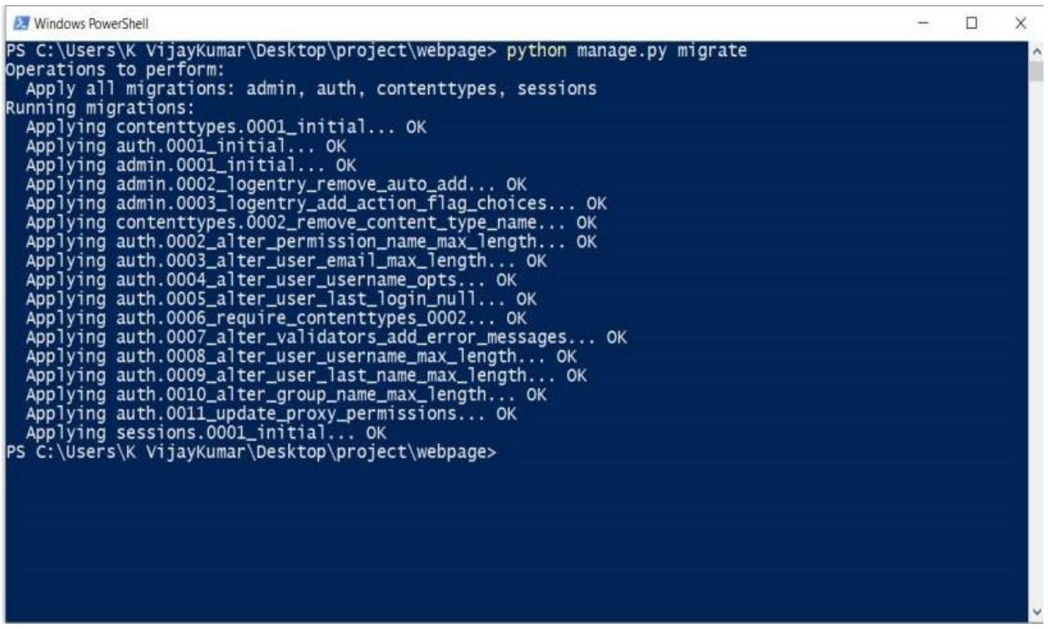


Windows PowerShell

```
PS C:\Users\K VijayKumar\Desktop\project\webpage> django-admin startapp disaster
PS C:\Users\K VijayKumar\Desktop\project\webpage>
```

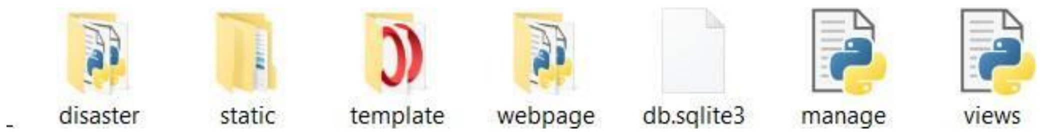
By the following command, we can store tables in the inbuilt database. (SQLite3)

➢ python manage.py migrate

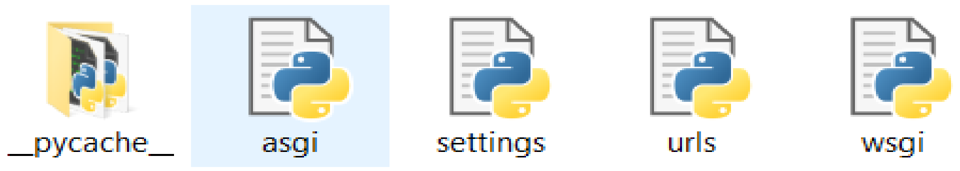


```
PS C:\Users\K VijayKumar\Desktop\project\webpage> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
PS C:\Users\K VijayKumar\Desktop\project\webpage>
```

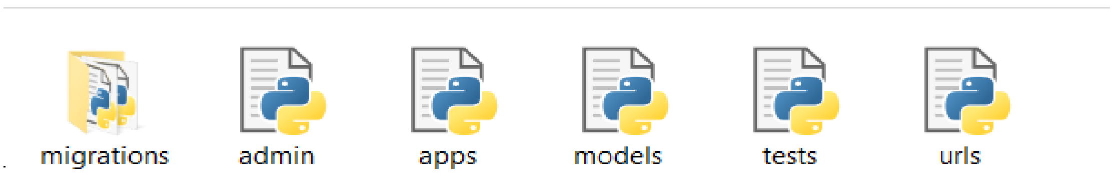
Project Files



Project files (webpage)

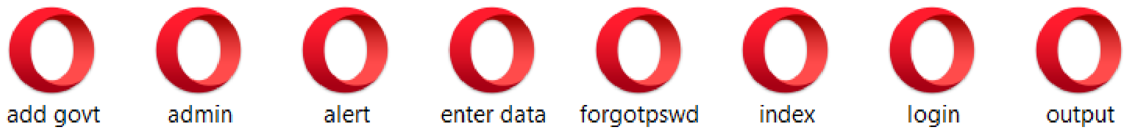


App files (disaster)



migrations admin apps models tests urls

Html files



add govt admin alert enter data forgotpswd index login output

6.1 views.py

```
from django.shortcuts import render
from disaster.models import main,login,givealert
from django.http import HttpResponse

def index(request):
        Template_get="index.ht
        ml"                     if
        request.method=='GET'
        :
                govtalert={'alert':givealert.objects.last()}
                return
                render(request,Template_get,govtalert)
        if                      request.method=='POST':
                govtalert={'alert':givealert.objects.last()}
                return
                render(request,Template_get,govtalert)

def index_post(request):
        Template_post="output.ht
        ml"                     if
        request.method=='POST':
                try:
                        num=request.POST.get('input',None)
                        selected_choice=main.objects.get(yourlocation=n
                        um) context={'data':selected_choice}
                except main.DoesNotExist:
                        raise           Http404("login
                DoesNotExist")                  return
                render(request,Template_post,context)

def govt_page(request):
        Template_alert="alert.html
        "     Template="login.html"
        Template_admin="admin.ht
        ml"  if  request.method  ==
        'GET':
                return render(request,Template)

        if request.method == 'POST':
                user                            =
                request.POST.get('user',False) pas =
                request.POST.get('passkey',False)
                if  login.objects.filter(username=user).exists()
and login.objects.filter(password=pas).exists():
```

```
                        return render(request,Template_admin)
                else:
                        context={'error':"username    or    password    is
                        wrong"}                                          return
                        render(request,Template,context)
def forgot(request):
        Template="forgotpswd.ht
        ml" if request.method ==
        'GET':
                return render(request,Template)

        if request.method == 'POST':
                username                                     =
                request.POST.get('username',False)
                schoolans                                    =
                request.POST.get('school',False)     user     =
                login.objects.get(username=username)    ans
                = login.objects.get(answer=schoolans)
                if user and ans:
                        context={'user':login.objects.get(username=userna
                        me)} return render(request,Template,context)

def adddata(request):
        if       request.method        =="GET":
                Template_alert            ="enter
                data.html"
                return render(request,Template_alert)

        if    request.method    ==    "POST":
                Template_data="enter
                data.html"

                safeplaces1    =    request.POST.get('safeplace1')
                safeplaces2    =    request.POST.get('safeplace2')
                safeplaces3    =    request.POST.get('safeplace3')
                safeplaces4    =    request.POST.get('safeplace4')
                safeplaces5    =    request.POST.get('safeplace5')

                transports1    =    request.POST.get('transport1')
                transports2    =    request.POST.get('transport2')
                transports3    =    request.POST.get('transport3')

                rescue1 = request.POST.get('rescue1')
                rescue2 = request.POST.get('rescue2')
                rescue3 = request.POST.get('rescue3')
```

```
            if      main.objects.filter(yourlocation=safeplaces1).exists():
                    context={'error':"entered location is all ready exist"}
                    return render(request,Template_data,context)

            else:
                    post=main(yourlocation=safeplaces1,
                        safeplacesname=safeplaces2,
                        safeplacesdistance=safeplaces3,
                        safeplacescapacity=safeplaces4,
                        safeplacescontact=safeplaces5,
                        transportname=transports1,
                        transporttypeof=transports2,
                        transportcontact=transports3,
                        rescueteammembers=rescue2,
                        rescuecontact=rescue3,
                        rescueteamtype=rescue1,
                        )
                    post.save()
                    return render(request,Template_data)

def govtadmin(request):
        Template_addgovt="add   govt.html"
        return
        render(request,Template_addgovt)
def alert(request):
        Template_admin="alert.ht
        ml"                        if
        request.method=="GET":
                return
        render(request,Template_admin)          if
        request.method=="POST":
                alert = request.POST.get('alert_','True')
                instance    =    givealert.objects.create(alertinfo=alert)
                return render(request,Template_admin)
```

6.2 Urls.py (disaster)

```
from  django.urls  import  path
from    django.contrib    import
admin
from views import govt_page,alert,adddata,govtadmin,index,index_post,forgot

urlpatterns    =    [
    path('', index,),
    path('main/',index_post,),
   path('login/',govt_page,name="govtlogin"),
   path('login/forgotpassword',forgot),
   path('login/alert/',alert,name="alert"),
   path('login/newdata/',adddata,name="newdata"),
   path('login/addgovt/',govtadmin,name="addgovt")
]
```

6.3 Urls.py (webpage)

```
from      django.contrib      import
admin  from  django.urls  import
include,path import views

urlpatterns = [
    path('admin/',    admin.site.urls),
    path('',include('disaster.urls')),
]
```

6.4 Manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative
tasks.""" import os
import sys


def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'webpage.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did
            you " "forget to activate a virtual environment?"
        )
    execute_from_command_line(sys.argv)


if__name___== '_main_':
    main()
```

6.5 Settings.py

```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(_file_)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY =
'k4j=g_2hx44+vgv1n_)3yby04=x73ouuis2ezemmlz&^@h@#@o'

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

```
DEBUG = True

ALLOWED_HOSTS = []
# Application definition

INSTALLED_APPS = [
    'disaster.apps.DisasterConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

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                    =
'webpage.urls' TEMPLATES = [
    {
        'BACKEND':
        'django.template.backends.django.DjangoTemplates',      'DIRS':
        ["template"],
        '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',
            ],
        },
    },
]
```

```
STATICFILES_DIRS = [
   os.path.join(BASE_DIR,                  "static"),
   '/webpage/static',
]

WSGI_APPLICATION = 'webpage.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
   'default': {
      'ENGINE':  'django.db.backends.sqlite3',  'NAME':
      os.path.join(BASE_DIR, 'db.sqlite3'),
   }
}



# Password validation
# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

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

https://docs.djangoproject.com/en/3.0/topics/i18

n/ LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'
```

```
USE_I18N = True
USE_L10N  =  True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
#   https://docs.djangoproject.com/en/3.0/howto/static-

files/ STATIC_URL = '/static/'
```

6.6 Apps.py

```
from django.apps import
AppConfig class
DisasterConfig(AppConfig):
    name = 'disaster'
```

6.7 Asgi.py

```
import os

from django.core.asgi import get_asgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'webpage.settings')
application = get_asgi_application()
```

6.8 wsgi.py

```
import os

from django.core.wsgi import get_wsgi_application
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'webpage.settings')
application = get_wsgi_application()
```

6.9 models.py

```
from django.db import
models class
main(models.Model):
#safeplaces
    yourlocation =
    models.CharField(max_length=200,default='null')
    safeplacesname =
    models.CharField(max_length=200,default='kkd')
    safeplacesdistance = models.IntegerField(default='23')
    safeplacescapacity = models.IntegerField(default='200')
    safeplacescontact = models.IntegerField(default='5679')

#transport
    transportname = models.CharField(max_length=200,default='orange
    travels') transporttypeof =
    models.CharField(max_length=200,default='bus') transportcontact =
    models.IntegerField(default='5679')

#rescue
    rescueteamtype =
    models.CharField(max_length=200,default='navy')
    rescueteammembers = models.IntegerField(default='5')
    rescuecontact =models.IntegerField(default='420420')

#hospital
class login(models.Model):
    question =
    models.CharField(max_length=100,default="testing")
    answer =
    models.CharField(max_length=100,default="testing")
    username=models.CharField(max_length=200,default='null')
    password=models.CharField(max_length=200,default='null')

#alert
class givealert(models.Model):
    alertinfo = models.CharField(max_length=50,default="testing")
```

6.10 admins.py

```
from django.contrib import admin
from .models import main,login,givealert

admin.site.register(main)
admin.site.register(login)
admin.site.register(givealert)
```

6.11 HTML Files:

index.html

```
<html>
<head>
<title>disaster management</title>
        {% load static %}
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<link rel="stylesheet"  href="{% static 'css/main.css' %}" type="text/css">
</head>
<body bgcolor="grey">
<div class="disaster">
<h3 class="disastercenter">DISASTER MANAGEMENT</h3></div>
<form method="post" action="">
{% csrf_token %}

<h2 class="alertmain">ALERT:</h2><label class="alertindex">{{ alert.alertinfo
}}</label><br>
  <input type="submit" value="update" class="alertupdate" />
</form>
<br>

<form method="POST" action="/main/">
{% csrf_token %}
<label class="labelindex">Enter your location:</label><br>
<input type="text" placeholder="enter location" name="input" class="input"></input>
<button class="inputbutton">submit</button>
</form>

<label class="govlabel">govt offical</label><br>
<img src="{% static 'css/govt.png' %}" class="govtmain"><br>
<input type="button" onclick="location.href='{% url 'govtlogin' %}'"
class="logingov" name="govtlogin" value="log in">

<form method="POST" action="/admin/">
{% csrf_token %}
<label class="djangoadmin">admin</label><br>
<img src="{% static 'css/govt.png' %}" class="govindex"><br>
<button class="govtlogin">login</button>
</form>

<div class="refabout">
<label class="refrence">Refrence Link</label><br>
```

```
<label class="link1">1st Link:</label> <a class="reflink"
href="https://www.msc.fema.gov/portal/search#searchresultsanchor">https://msc.fema.g
ov/p ortal/search#searchresultsanchor</a>
<label class="link2" >2nd Link:</label> <a class="reflink1"
href="https://www.Disaster- survival-resources.com">www.Disaster-survival-
resources.com"</a>
<label class="link3">3rd Link:</label> <a class="reflink2"
href="https://www.media.ifrc.org/ifrc">www.media.ifrc.org/ifrc
</a>

<label class="about">About us</label>
<label class="about1">admn name:hitler</label>
<label class="about2">contact:123456</label>
<label class="about3">email:hitler@gmail.com</label>
<label class="about4">address:india</label>

</div>
</body>
</html>
```

Output.html

```
<html>
<head>
<title>main</title>
</head>
{% load static %}
<link rel="stylesheet" href="{% static 'css/main.css' %}" type="text/css">

<body bgcolor="grey">
<div class="output">
<center><label class="outputloc">Enter your
location:{{data.yourlocation}}</label></center>
<br>
<center><table border="2">
<tr>
<th colspan="2"><center>Safeplaces</center></th></tr>
<tr>
<td><center>safepoint:</center></td><td><center>{{data.safeplacesname}}</center></td>
</tr>
<tr><td><center>distance:</center></td><td><center>{{ data.safeplacesdistance
}}</center></td></tr>
<tr><td><center>contact:</center></td><td><center>{{ data.safeplacescontact
}}</center></td></tr>
<tr><td><center>capacity:</center></td></td><td><center>{{ data.safeplacescapacity
}}</center></td></tr>
<tr>
<th colspan="2">Transport</th></tr>
<tr>
<td><center>name</:</td><td><center>{{ data.transportname }}</center></td></tr>
<tr><td><center>type of:</center></td><td><center>{{data.transporttypeof
}}</center></td></tr>
<tr><td><center>contact</:</td><td><center>{{ data.transportcontact
}}</center></td></tr>
<tr><th colspan="2">Rescue team</th></tr>
<tr><td><center>team members:</center></td><td><center>{{ data.rescueteamtype
}}</center></td></tr>
<tr><td><center>team type:</center></td><td><center>{{ data.rescueteammembers
}}</center></td></tr>
<tr><td><center>team contact:</center></td><td><center>{{ data.rescuecontact
}}</center></td></tr>
</table></center>
</div>
</body></html>
```

Login.html

```
<html>
<head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
{% load static %}
<link rel="stylesheet" href="{% static 'css/main.css' %}" type="text/css">
<title>login</title>
</head>
<body bgcolor="grey">
<img src="{% static 'css/govt.png' %}" class="govtimage" />
<center><div class="title"><h2>govt offical</h2></div></center>
<form method="POST" action="">
        {% csrf_token %}
        <h3 class="errorlogin">{{error}}</h3>
<label class="login"><b>Login:</b></label><input type="text" placeholder=" enter login"
name="user" class="logintext"></input><br>
<label class="password"><b>Password:</b></label><input type="password"
placeholder="enter password" name="passkey"
class="passwordtext"></input><br>
<a href="forgotpassword" class="forgotpassword1">forgot password</a>
<button class="loginbutton1">login</button>
</form>
</body>
</html>
```

Admin.html

```
<html>
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   {% load static %}
<link rel="stylesheet" href="{% static 'css/main.css' %}" type="text/css">
<script src="{% static 'css/test.js' %}" type="text/javascript"></script>
<title>govt user</title>
<body bgcolor="grey">
<center><h1 class="admintitle">govt officals</h1></center><br>
<div class="border">
<label class="welcome">welcome user</label><br><br>
<form method="POST" action="alert/">
         {% csrf_token %}
<button onclick="location.href='{% url 'alert' %}'" name="username"
class="alert" placeholder="give alert" value="enter alert" >enter
alert</button><br><br>
</form>
<form method="POST" action="newdata/">
         {% csrf_token %}
<input type="button" name="username" onclick="location.href='{% url 'newdata'
%}'" class="newdata" placeholder="give alert" value="add new data"/><br><br>
</form>
</div>
</body>
</html>
```

Forgot password.html

```
<html>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
        {% load static %}
<link rel="stylesheet" href="{% static 'css/main.css' %}" type="text/css">
<body bgcolor="grey">
<form method="post">
                    {%csrf_token%}
<center class="forgottitle">FORGOT PASSWORD</center>
<center><label class="forgotuser">what is your user name:</label><input
type="text" placeholder="enter user name" name="username"
class="forgotusertext"><br></center>
<center><label class="forgotschool">what is your first school
name:</label><input type="text" placeholder="enter security answer"
class="forgotschooltext" name="school"><br></center>
<center><input type="submit" name="" class="forgotsubmit"></center>
<h2 class="usernameforgot">username:{{user.username}}</h2>
<h2 class="passwordforgot">password:{{user.password}}</h2>
</form>
</body>
</html>
```

Alert.html

```
<html>
<head>
        <title>send alerts</title>
        {% load static %}
 <link rel="stylesheet" href="{% static 'css/main.css' %}" type="text/css">
</head>
<body bgcolor="grey">
<form method="post">
        {%csrf_token%}
<label class="alertlabel">Enter alert/update alert</label>
<input type="text" placeholder="enter alert " class="alerttext" name="alert_"><br><br>
<input type="submit" class="alertbutton">
</form>
</body>
</html>
```

Enter data.html

```
<html>
<head>
<title>,enter new data</title>
          {% load static %}
<link rel="stylesheet" href="{% static 'css/main.css' %}" type="text/css">
<link rel="stylesheet" type="text/css" href="sample.css">
</head>
<body bgcolor="grey">
<center><h1 class="titl"><b>Enter data</b></h1></center>
<form method="post">
                    {% csrf_token %}
<div class="safeplace">
<h3 class="errordata">{{error}}</h3>
<b class="bold">safe places:</b><br><br>
<label class="enterlabel">location: </label><input type="text" placeholder="entered
location " class="safe" name="safeplace1" ></input><br><br>
<label class="enterlabel">safepoint: </label><input type="text"
placeholder="enter safepoint" class="safe" name="safeplace2" ><br><br>
<label class="enterlabel">distance: </label><input type="text" placeholder="enter
distance" class="safe" name="safeplace3"><br><br>
<label class="enterlabel">capacity: </label><input type="text" placeholder="enter
capacity" class="safe" name="safeplace4"><br><br>
<label class="enterlabel">contact: </label><input type="text" placeholder="enter
scontact" class="safe" name="safeplace5"><br><br>
</div>
<div class="transportfac">
<b class="bold">Transport:</b><br><br>
<label class="enterlabel">name: </label><input type="text"
placeholder="enter transportname" class="transport"
name="transport1"><br><br>
<label class="enterlabel">type: </label><input type="text" placeholder="enter
type" class="transport" name="transport2"><br><br>
<label class="enterlabel">contact: </label><input type="text" placeholder="enter
contact" class="transport" name="transport3"><br><br>
</div>
<div class="rescue">
<b class="bold">rescue team:</b><br><br>
<label class="enterlabel">teamtype: </label><input type="text"
placeholder="enter teamtype" class="rescueforce" name="rescue1" ><br><br>
<label class="enterlabel">members: </label><input type="text"
placeholder="enter members" class="rescueforce" name="rescue2"><br><br>
<label class="enterlabel">contact: </label><input type="text" placeholder="enter
contact" class="rescueforce" name="rescue3">
```

```
</div>
<input type="submit" value="submit" name="" class="enterdata">
</form>
</body>
</html>
```

7. TEST CASES

7.1 Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable a specific testing requirement.

7.2 Types of Testing

7.2.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is the structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contain clearly defined inputs and expected inputs and expected results.

7.2.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction as show by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

7.2.3 Functional Testing:

Functional test provide systematic demonstration that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must
be accepted Invalid Input	: identified classes of valid input must
be rejected Functions	: identified functions must be exercised

Output : identified classes of application outputs must be exercised

Systems/Procedures: Interfacing system is must be invoked

Organization and preparation of functional test is focused on requirements. Key functions are special test cases. In addition systematic coverage pertaining to identify business process flows; data fields, pre defined processes and successive processes must be considered for testing.

7.2.4. System Testing:

System testing ensures that the entire integrated software system meets requirements. It tests configurations to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process description and flows, emphasizing pre-driven process links and integration points.

7.2.4.1 White Box Testing:

White Box Testing is a testing in which the software tester has knowledge of the inner workings. Structure and language of the software are at least its purpose it is used to test areas that cannot reached from a black box level.

7.2.4 Black Box Testing

Black box Testing is testing the software without any knowledge of the inner working, structure or language of the module being tested. Black box tests, as most other kinds of tests. Black box testing is testing the software without any knowledge of the inner working structure or language of the module being tested. Black box tests, as most other kinds of tests must be written from the definitive source document, such as specification or requirement document. It is a testing in which the software under test is treated, as a black box. You cannot see into it. The test provides inputs and responds to outputs without considering how the software works.

7.3 Test Cases:

Unit Testing:

Unit testing is usually conducted as part of a combine and unit test phase of the software life cycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach:

Field testing will be performed manually and functional tests will be written in detail.

Test Objectives:

- All field entries must work properly.
- Pages must be activated from the identified links.
- The entry screen, messages and response must not be delayed.

Features to be tested:

- Verify that the entries are of the correct format.
- No duplicate entries should be allowed.
- All links should take the user to the correct page.

Integration Testing:

Software integration testing is a incremental integration testing of two or more integrated software component on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e .g components in a software system or –one step up –software applications at company level- interact without error.

Test Result:

All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing:

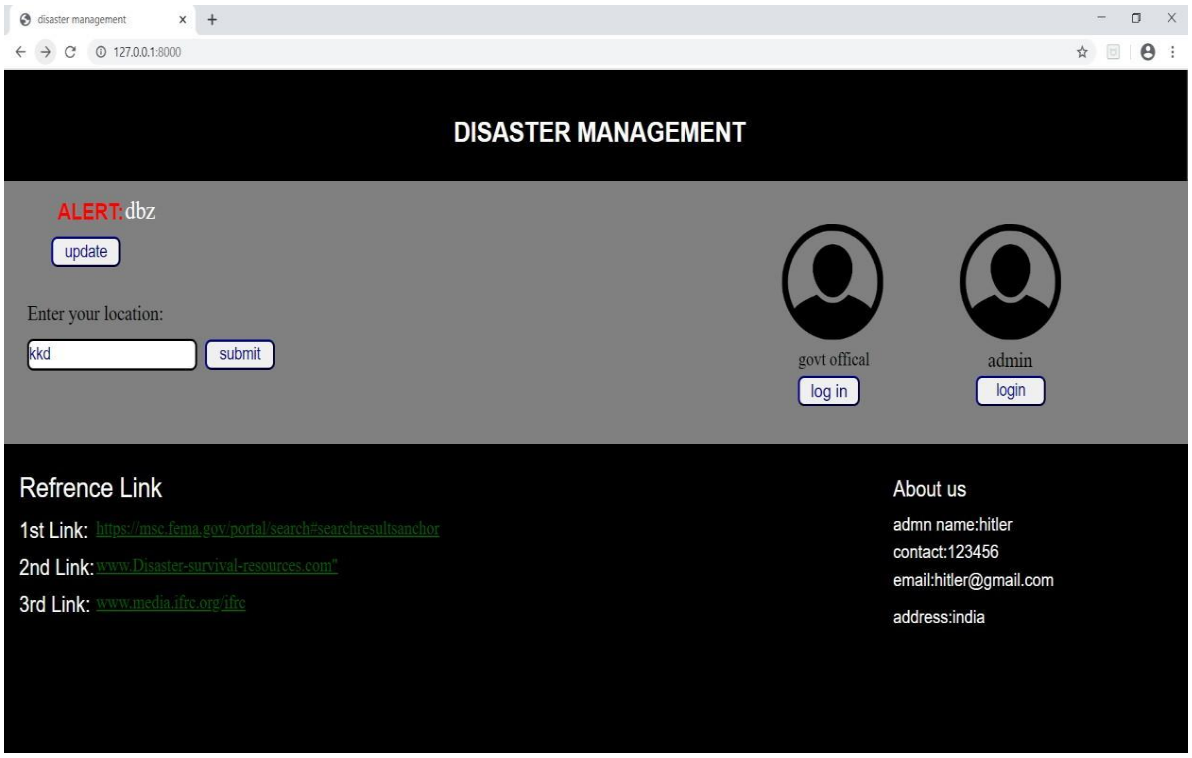
User acceptance testing is a critical phase of any project and requires significant participation by the end users. It also ensures that system meets the functional requirements.

TEST CASES ID	INPUT	DESCRIPTION	EXPECTED RESULT	ACTUAL RESULT	PASS/FAIL
OSA-1-01	Blank Password Blank	A blank username and password are required	Display user name or password is wrong	Display user name or password is wrong	Pass
OSA-1-02	Valid username Password Blank	A valid username and blank password is given by the user	Display user name or password is wrong	Display user name or password is wrong	Pass
OSA-1-03	User name blank valid and password	A blank username is valid password is given by the user	Display user name or password is wrong	Display user name or password is wrong	Pass

A valid username and valid password is given by the user	OSA-1-04	Valid user name valid passwor d	Login successful ly	Login successful ly	Pass
A username with numbers and valid password is given by the user	OSA-1-05	Username with numbers with valid password	Display user name or password is wrong	Display user name or password is wrong	Pass
A username with special symbol and valid password is given by the user	OSA-1-06	Username with %,$,#, valid password	Display user name or password is wrong	Display user name or password is wrong	Pass
An username with spaces and valid password is given by the user	OSA-1-07	Username with spaces valid password	Display user name or password is wrong	Display user name or password is wrong	Pass
An invalid username and invalid password is given by the user	OSA-1-08	Invalid userna me, invalid passwor d	Display user name or password is wrong	Display user name or password is wrong	Pass

8. OUTPUT SCREENS

Fig 8.1 Index Page:



disaster management

127.0.0.1:8000

DISASTER MANAGEMENT

ALERT:dbz

update

Enter your location:

kkd

submit

govt offical

log in

admin

login

Refrence Link

1st Link: https://msc.fema.gov/portal/search#searchresultsanchor

2nd Link: www.Disaster-survival-resources.com"

3rd Link: www.media.ifrc.org/ifrc

About us

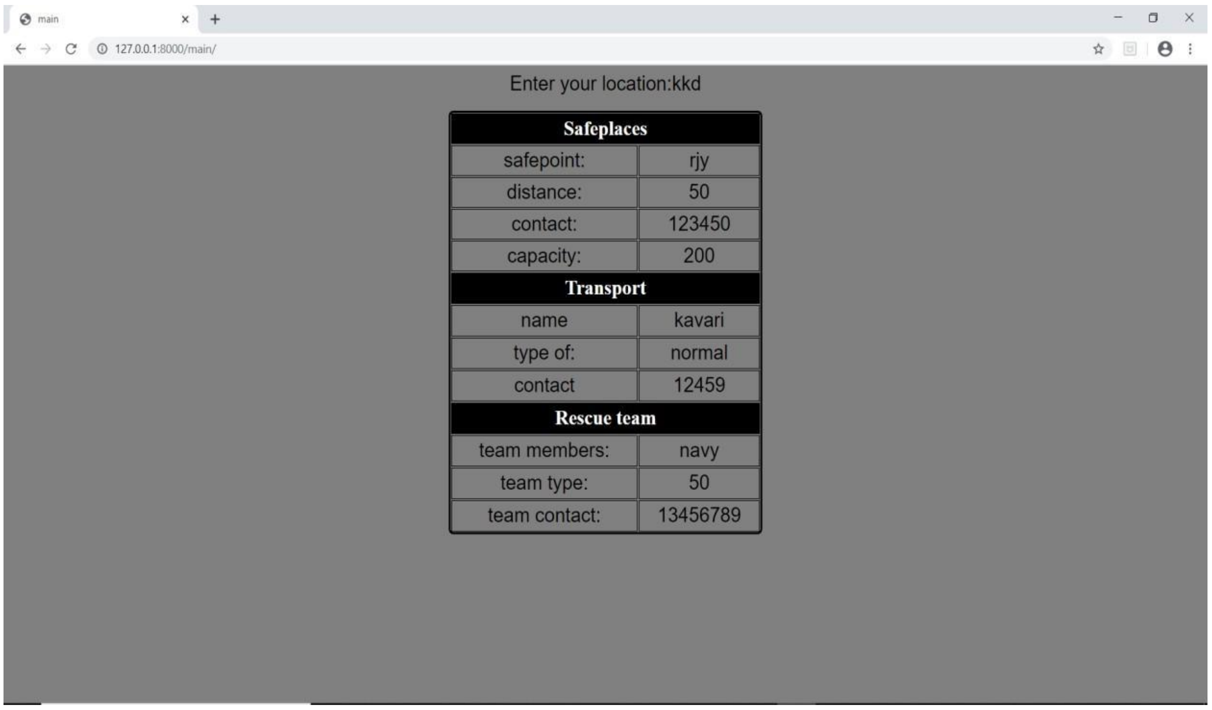
admn name:hitler

contact:123456

email:hitler@gmail.com

address:india

Fig 8.2 Safe Places:



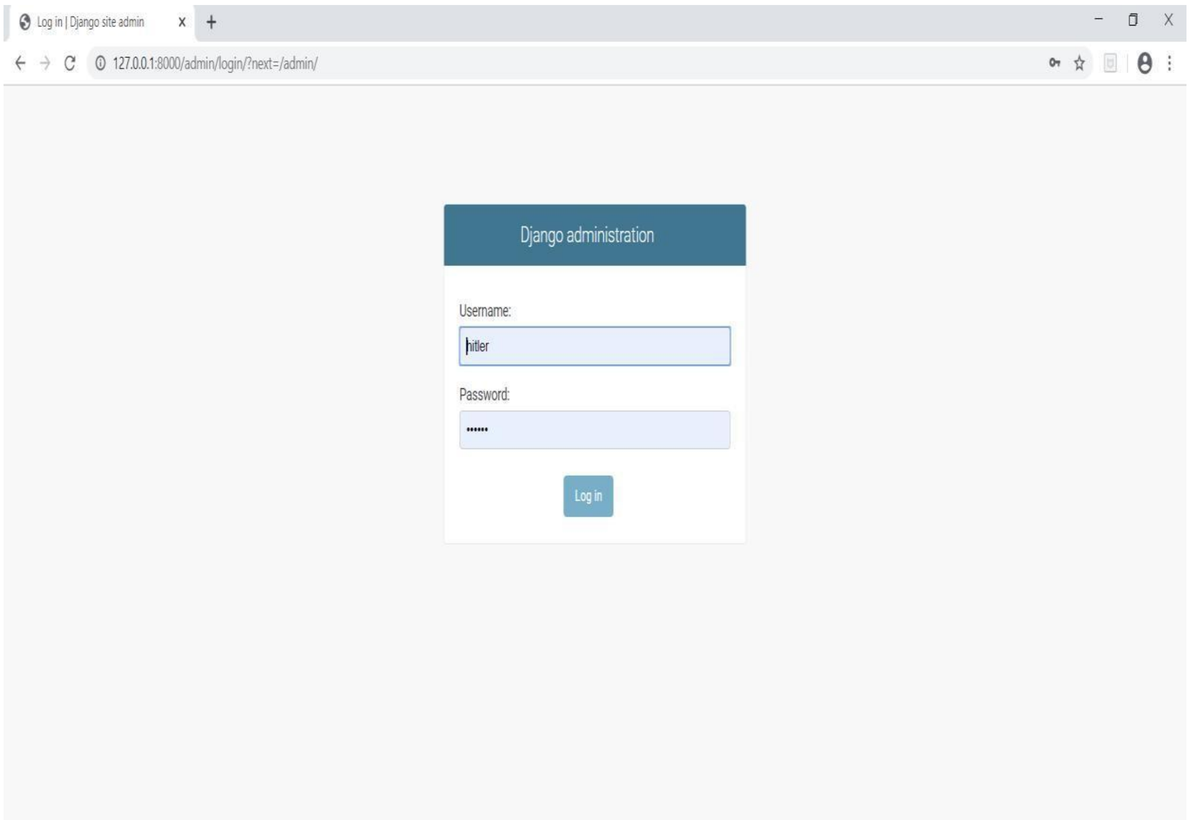
main

127.0.0.1:8000/main/

Enter your location:kkd

Safeplaces	
safepoint:	rjy
distance:	50
contact:	123450
capacity:	200
Transport	
name	kavari
type of:	normal
contact	12459
Rescue team	
team members:	navy
team type:	50
team contact:	13456789

Fig 8.3 Admin Login:



Log in | Django site admin

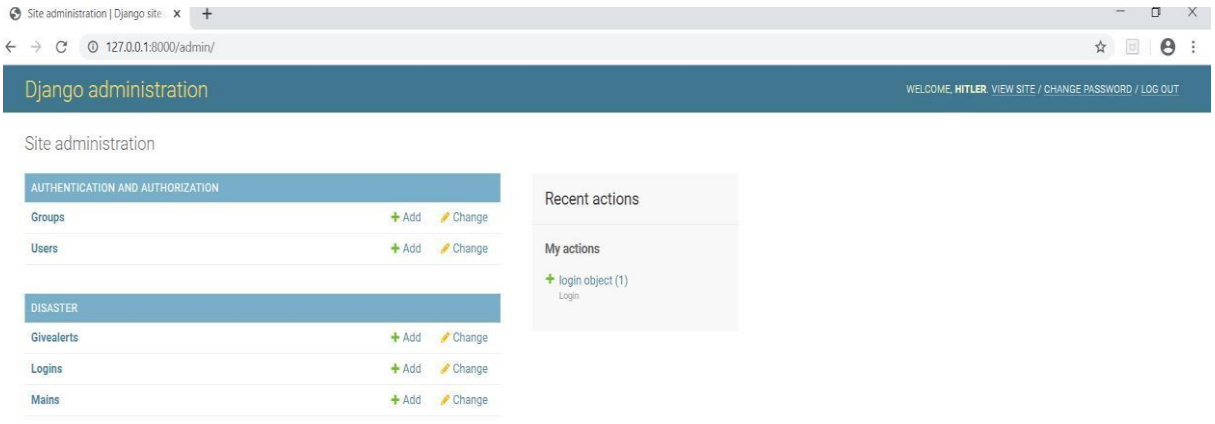
127.0.0.1:8000/admin/login/?next=/admin/

Django administration

Username: hitler

Password: ••••••

Fig 8.4 Administration Site:



Django administration

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

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

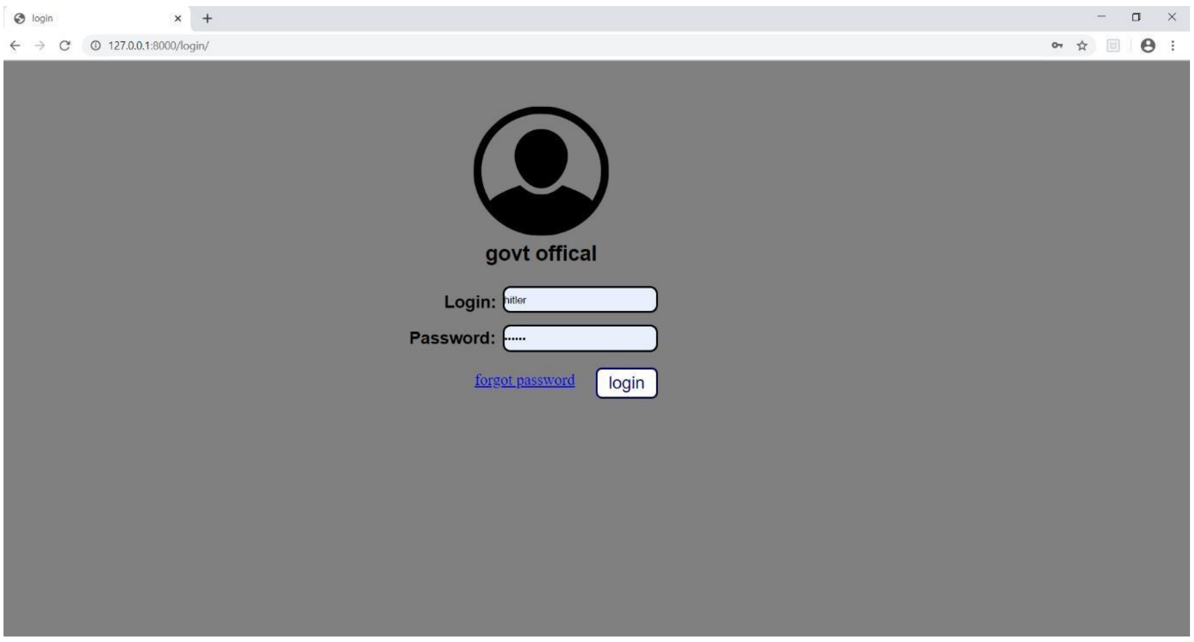
DISASTER		
Givealerts	+ Add	Change
Logins	+ Add	Change
Mains	+ Add	Change

Recent actions

My actions

+ login object (1)
Login

Fig 8.5 Government Official Login:

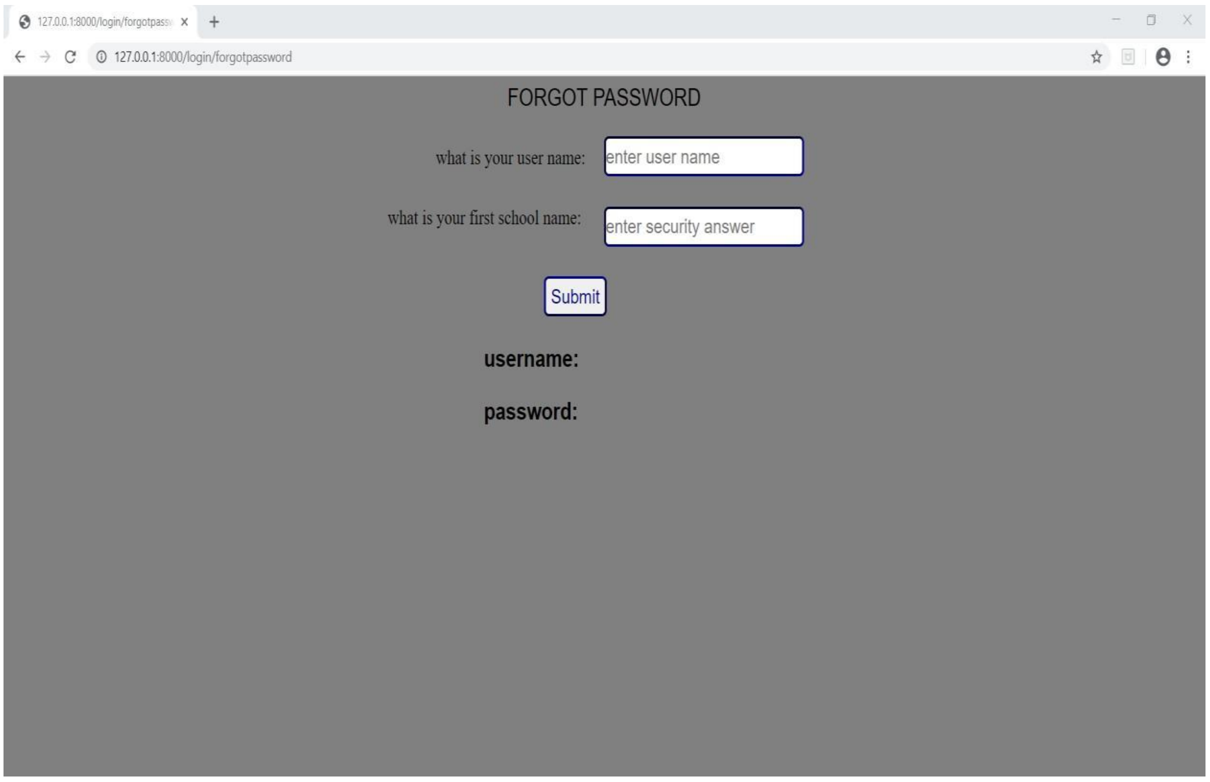


govt offical

Login: hitler

Password: ......

Fig 8.6 Forgot Password:


127.0.0.1:8000/login/forgotpass

127.0.0.1:8000/login/forgotpassword

FORGOT PASSWORD

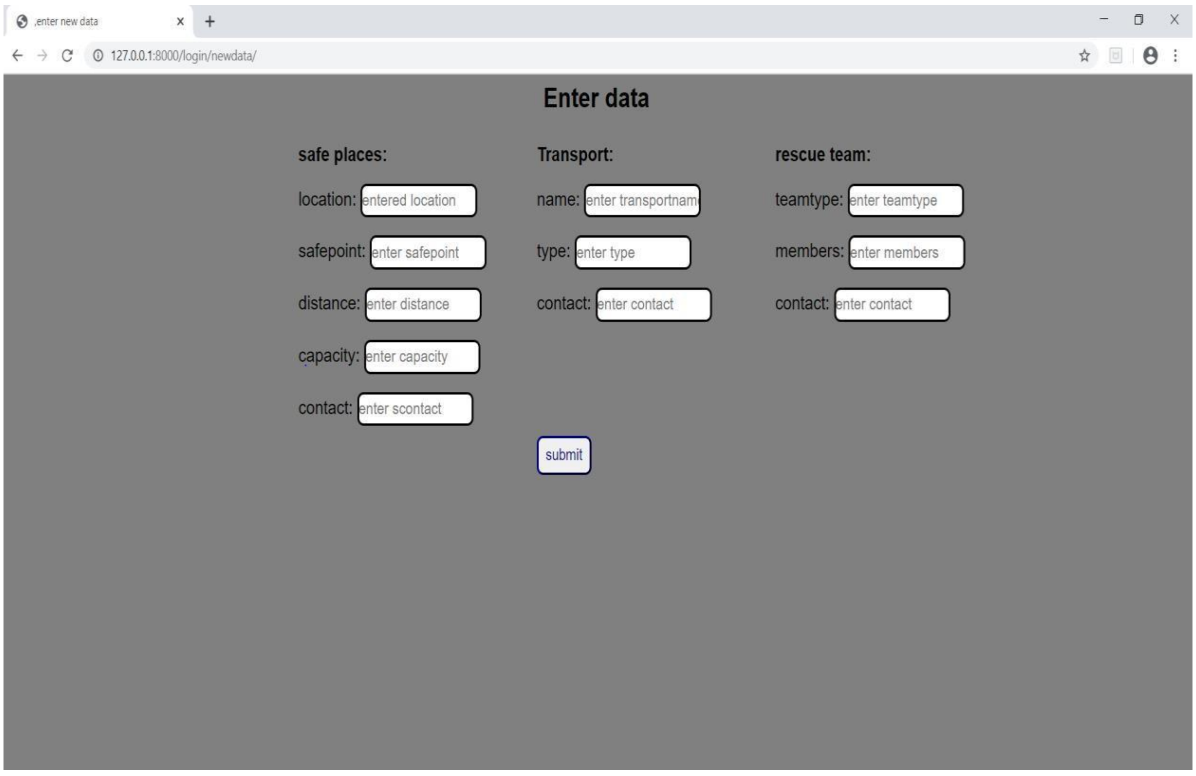
what is your user name: enter user name

what is your first school name: enter security answer

username:

password:

Fig 8.7 Adding New data:


,enter new data

127.0.0.1:8000/login/newdata/

Enter data

safe places:	Transport:	rescue team:
location: entered location	name: enter transportnam	teamtype: enter teamtype
safepoint: enter safepoint	type: enter type	members: enter members
distance: enter distance	contact: enter contact	contact: enter contact
capacity: enter capacity		
contact: enter scontact		

9. CONCLUSION AND FUTURE SCOPE

Disaster Management on being implemented shows the nearby safe places, alerts about floods and other details like transport and rescue.

This project is good in saving the life of victims in the time of floods. It will be more useful when you visit the place which you don’t know previously. We know that nothing is more precious than life of us. This project helps in surviving the lives of people in the above mentioned situation. We used Django technology in it for building this website. It contains inbuilt database named sql lite3, so we can use that database for the needs of database. Adding new data to this database is easy task when compared to other databases. We has given the duty of entering details about safe places to government as we can get the correct information. User will get the safe place details by entering his current location.

The world will not look the same as it does today. Shifting demographics and the rate of technological innovation will challenge the way we plan and communicate with the public. More frequent and more intense floods will present operational challenges and complexities. Any of these issues alone would challenge some current emergency management policies and procedures. Forces of change produce a difficult, highly uncertain future, the complexity of which will test the ability of the emergency management community to execute our mission. Exploring the nature of such future challenges can help us take actions to improve resilience and adaptability. Rapid advancement of technology could be deployed in efficiently tackling the challenges emerging from disasters, minimizing the impact of disasters in terms of reducing the magnitude of death and casualties, improving the health and sanitary conditions of the affected population, rehabilitation of the victims, etc. Specific technological solutions can be utilized in all the phases of disaster management, namely, disaster preparedness, disaster reduction, disaster mitigation, and post- disaster rehabilitation. There is a need for the application of modern technologies in disaster management, wherever and whenever possible. Many frontier areas such as space technology, modern information and communication systems, renewable energy, advanced medical diagnostics, and remotely operated robotic systems for rescue and relief operations; find useful applications in disaster management efforts.

10. BIBILOGRAPHY

- ➢ Mastering Django: Core – The Complete Guide to Django 1.8 LTS
- ➢ Geeks for Geeks
- ➢ Stack Overflow open source platform

10.1 REFERENCES AND WEBSITE REFERENCE:

- https://www.cbc.ca/news/world/the-world-s-worst-natural-disasters-1.743208 The world's worst natural disasters Calamities of the 20th and 21st centuries] CBC News Retrieved 2010-2-10
- http://www.pbs.org/wgbh/nova/flood/deluge.html
- http://www.time.com/time/specials/packages/article/0,28804,1953425_1953424,00.html
- http://earthquake.usgs.gov/earthquakes/eqinthenews/2010/us2010rja6/#summary
- https://link.springer.com/chapter/10.1007/978-94-017-0137-2_6
- Object Oriented Analysis and Design with application by Grady Booch
- Fundamentals of python by Reema Thereza
- Introduction to Django by Dr. R. Nageswara Rao
- https://docs.djangoproject.com/en/3.0/