

# **USER MANAGEMENT SYSTEM**

Project report submitted in partial fulfilment of the requirement for the  
degree of Bachelor of Technology  
in

**Computer Science and Engineering/Information Technology**

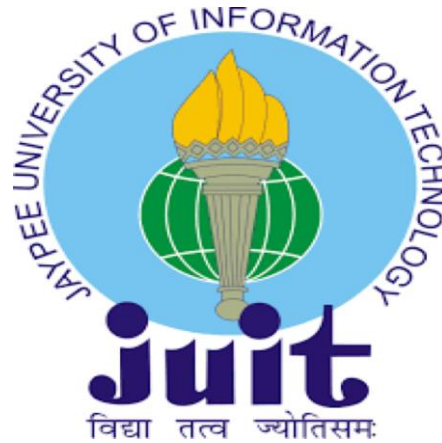
By

SAHIL KAITH (161247)

Under the supervision of

Dr. Hari Singh Rawat

to



Department of Computer Science & Engineering and Information  
Technology

**Jaypee University of Information Technology Waknaghat, Solan-  
173234, Himachal Pradesh  
CERTIFICATE**

**Candidate's Declaration**

I hereby declare that the work presented in this report entitled “**User Management System**” in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from 23, May 2020 to 22, June 2020 under the supervision of Dr. Hari Singh , Assistant Professor (SG) , Department of Computer Science & Engineering and Information Technology. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Sahil Kaith  
Roll No - 161247

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Dr. Hari Singh  
Assistant Professor (SG)  
Department of Computer Science  
& Engineering and IT  
Dated:

## ACKNOWLEDGEMENT

I would like to express my gratitude to my mentor Dr. Hari Singh **who** guided me throughout the phase of making project (“**User Management System**”). He helped me in the thorough understanding of fundamentals of project, understanding the basic functionalities, implementing the functionalities and making my project more responsive and much efficient. I would also like to extend my heartiest thanks to all members of JUIT for their support and guidance throughout my project.

Secondly, I would also like to thank my friends who helped me in completing and finalizing project in given time.

## TABLE OF CONTENTS

Page	TABLE OF CONTENTS
6	LIST OF FIGURES
7	ABSTRACT
8	1.INTRODUCTION
8	1.1 INTRODUCTION
10	1.2 PROBLEM STATEMENT
11	1.3 OBJECTIVE
12	1.4 METHODOLOGY
13	1.5 ORGANIZATION OF PROJECT
14	2. LITERATURE SURVEY
14	2.1 DJANGO AUTHENTICATION SYSTEM
18	2.2 DJANGO USER MANAGEMENT
20	3.SYSTEM DEVELOPMENT
20	3.1 DESCRIPTION OF PROJECT
23	3.2 DESIGN DIAGRAM
25	3.3 FUNTIONAL & NON-FUNTIONAL REQUIRE
27	3.4 IMPLEMENTATION DETAILS
28	3.4.1 SETTING UP PROJECT
30	3.4.2 CREATING REQUIRED APPS
39	3.5 IMPLEMENTATION ISSUES

<b>Page</b>		<b>TABLE OF CONTENTS</b>	
<b>40</b>		4.PERFORMANCE ANALYSIS	
<b>40</b>		4.1 AGILE METHODOLOGY	
<b>41</b>		4.2 ANALYTICS	
<b>42</b>		4.3 OUTPUTS	
<b>47</b>		5.CONCLUSION	
<b>47</b>		5.1 CONCLUSIONS	
<b>48</b>		5.2 FUTURE SCOPE	
<b>49</b>		5.3 REFRENCES	

## LIST OF FIGURES

FIGURE NUMBER	CONTENTS
2.1	Default user creation
2.2	Password reset
2.3	Authenticate user
2.4	Authenticate user in web request
3.1	URL used
3.2	Use case diagram
3.3	ER - Diagram
3.4	Django Architecture
3.5	Root folder of the project
3.6	Installed apps
3.7	Views.py of accounts
3.8	Views.py of login
3.9	Views.py of logout
3.10	Views.py of home
4.1	Homepage
4.2	Login page
4.3	New user registration page

## **ABSTARCT**

This project aims at creating a simple user management system that is required by every website where multiple users can login. This let new user registration, login & logout for every user. Authentication is done every time a login attempt is made. The data of every user is stored in database, thus it make use of database fetch & store utility.

This project is a simple full stack project built with little bit of every tool. Python is used for programming, Django is used as a framework to support various application. The frontend work is basic & simple utilizing the tools. Adequate messages & popups are shown for very mistake and any restricted task.

Thus the management of users will help users access to the website. The validation of data in this project is quite strong. Unique data must be their for every user.

# CHAPTER - 1

## INTRODUCTION

### 1.1 Introduction

This project is about how the user data is stored in every website & application. How users are able to create new account and sign in based on the new registration.

Users are able to login based on their credentials and every time they log in, authentication is done, means the data in the data bases is matched with the data entered in the login form. The login form is submitted to Django view, password & username are matched, only if they match the website can be accessed, otherwise the site shows an error.

Similarly when a new user wants to register, he/she has to create a new account entering details including the username , full name, email, password & reenter password.

Even at this time the details entered by the user are matched with database to check for similar data, an error pops up if data is not unique, username & email should be unique.

The user is redirected to the same page in case an invalid attempt is made. For example , if an invalid data is entered in the registration form , the user is redirected again to fresh registration form .

Similarly in the login page, the user is redirected back to the login page in case of invalid data or if the username and password don't match the data in the database.

This project use of Python language for coding and Django for framework. Django being a high-level Python Web framework encourages rapid development and clean design.

For login page , a template is used to show the multiple feautres of the Django framework. The rest pages are made from scratch using HTML & CSS.



The content to the pages is kept to be minimum to just show the real work going on in the backend ,

In this project there are super users with special permissions to access to the database who only have control to all user data. No user can access this, nor a user can access other user data.

## **1.2 Problem Statement:**

The user management system is a vital part of the every website which can have any number of users, For example, in Facebook there are currently 2.6 billion users currently.

The data of these users needs to be stored in the database to let them access Facebook based on their credential details.

Similarly, there are many other websites with large number of users. But even for websites with small number of users , User management system is needed.

For example financial websites, Online banking websites, government sites, military sites and more. The data of users in such sites can be a hectic thing to account for.

The most important factor that comes is the security & authentication of this user data. User data is the most commonly sold thing over dark-internet. User credentials are easily tempered with and used for ill purposes.

Safeguarding this data is also an important part of User management system. For this database needs to be well protected.

Every website have data which should be accessed with proper authorization. So, management of user is much needed. Most of the website allows users to access it only on login or when a user creates an account in that website.

### **1.3 Objective**

The main objective of this project is to help a random user get access to website. For this user needs login credentials, if user have them he can login with them and get access to site.

Else, user is given an option of creating new account entering his/her details and this details get saved in the database. Then user is redirected to login page to sign in with credentials.

Any number of users can create new account with unique data , get access to their individual data in the website. The objective of this is to separate the feed of website for every unique user.

## 1.4 Methodology

This project is the basic implementation of web framework in connection with database.

The project consist of mainly 3 pages in frontend

1. Homepage : where user can login, register. If not logged in, or logout if logged in.
2. Login : where a user can login based on his/her credentials.
3. New User Registration : where a new user can register.

The framework is controlled by Django, which take all inputs and works on them,

1. Authenticate the data.
2. Validate the data.
3. Project the frontend files.
4. Works with backend/database to access user data and store new data.

All the codes are written in Python. The conditions, functions and variables.

## **1.5 Organisation of report**

- Chapter 1 gives the basic introduction about project and basic fundamentals that will be used in implementing the project.
- Chapter 2 gives us the literature survey of “User Management System”. In this we will have brief of journals, research papers, internet source of application.
- Chapter 3 is most important chapter in which we will discuss about the implementation of project including explanation and source code of the project.
- Chapter 4 is the model used for analyzing the performance of User Management System. Outputs of the project at various stages and comparisons between different outputs.
- Chapter 5 will cover the end conclusion of the project and what further can be implemented in future to this project so that it becomes more efficient and reliable.

## CHAPTER 2

### LITERATURE SURVEY

#### 2.1 Django authentication system:

This system has been updated time and again to make sure it helps to cater the most common project needs, handling a wide range of methods. For projects which needs authentication needs different from the default configuration, Django supports extensive extension and customization per the growing varying needs of different websites.

It provides both authentication & authorization and is sometime called as authentication system, as these features are somewhat similar.

User objects:

User objects are the main element of the authentication system. They mainly represent the people who are interacting with the website and these user objects are used to activate method like restricting access, registering new user, allowing users to part of the creators etc. There is only one class of user that exists in Django's authentication framework called 'superusers' users and they are just default user objects with special attributes set, not so different than classes of user objects.

The default attributes are:

- username
- password
- email address
- first name
- last name

```

>>> from django.contrib.auth.models import User
>>> user = User.objects.create_user('john', 'lennon@thebeatles.com',
'johnpassword')

# At this point, user is a User object that has already been saved
# to the database. You can continue to change its attributes
# if you want to change other fields.
>>> user.last_name = 'Lennon'
>>> user.save()

```

Fig. 2.1 Default User Creation

Now creating super users :

We create super users using the createsuperuser command in shell command:

```

$ python manage.py createsuperuser --username=joe --email=joe@example.com

```

You will be asked for a password. After we enter it a new user will be created at that very instant. You can let go of username and email options, it will only prompt you for those values. Superuser are those users who controls the admin page of websites, where all data stored in the database can be accessed using models from Django framework.

Changing passwords :

Django does not save text passwords on the user model, but it stores only a hash this is because of this, we should not attempt to change the password attribute of the user directly in any condition. That is why a helper function is used when a user is created.

We can change user's password by many ways :

Manage.py change password \*username\* is good method of changing a user's password from the command line prompt . It asks you to change the password of the given user, which you

have to enter twice. If they happen to match, the new password will be changed with immediate effect.

You can also change a password in the program by using set password() method:

```
>>> from django.contrib.auth.models import User
>>> u = User.objects.get(username='john')
>>> u.set_password('new password')
>>> u.save()
```

Fig. 2.2 Password Reset

Command used to authenticate users :

```
authenticate(request,credentials)
```

We use authenticate() to verify credentials of a particular user. It takes login credentials as arguments, username & password, then matches them against each set of credentials in the backend and returns a User object if the credentials matches with the backend. If the credentials aren't valid for any backend or if a backend raises Permission Denied and returns None. For example:

```
from django.contrib.auth import authenticate
user = authenticate(username='john', password='secret')
if user is not None:
    # A backend authenticated the credentials
else:
    # No backend authenticated the credentials
```

Fig. 2.3 Authenticate User

Authentication in Web requests :

Django make use of sessions and middleware to implement the authentication system of user objects.



This is done by providing `request.user` attribute to current user who is logged in . If the current user has not logged in, this attribute will be automatically set to a object of anonymous User, otherwise it will be an instance of defined user.

You can distinguish between different type of users with `is_authenticated`, like so:

```
if request.user.is_authenticated:
    # Do something for authenticated users.
    ...
else:
    # Do something for anonymous users.
    ...
```

Fig. 2.4 Authentication in Web request

Default permissions :

Django.contrib.auth is a app which is listed in the `INSTALLED_APPS` in `setting.py` which provides default permissions, it ensures four default permissions which are:

- add
- change
- delete
- view

these are created for each Django user model.

These permissions are created when we run `manage.py migrate`;

The default permissions will be created for all previously made models, as well as for any new models which are being installed at that time. After that it will create default permissions for new models each time you run `manage.py migrate` .

However the Permission model is rarely accessed directly.

## 2.2 Django-user-management system [Incuna]

This User management system is built on Django and Django REST framework.

This allow to create, identify users (from their email-id instead of their username) as well as it allows sending password reset and account validation emails.

User management model give flexibility to create your own User model. This project allow to create, identify users (from their emails instead of their username) as well as sending password reset and account validation emails.

This User management system can be grouped into five sections:

- `auth`: authenticate and destroy a user session.
- `password_reset`: send and confirm a request to reset a password.
- `profile`: view current user profile.
- `register`: create a new account
- `users`: give a list and details of users.

Project includes `user_management.api.urls` will give the following methods:

- `auth`
- `password_reset`
- `profile`
- `register`

Url are as following:

Auth:

- url: `/auth`

Password reset:

- url: `/auth/password_reset/confirm/<uid>/<token>`
- url: `/auth/password_reset`

Profile:

- url: `/profile`

- url: /profile/password

Register:

- url: /register
- url: /resend-confirmation-email

Users:

- url: /users
- url: /users/<pk>

Verify email:

- url: /verify\_email/<uid>/<token>

Thus creating a strong validation of accounts and ensuring no duplicate accounts with single email and email-verification also let to validate the email entered

The login credentials are email & password.

Such a work inspire to create more robust user management system which can be used by websites.

## **CHAPTER 3**

### **SYSTEM DEVELOPMENT**

### **3.1 Description of user management system:**

Users are able to login based on their credentials and every time they log in, authentication is done, means the data in the data bases is matched with the data entered in the login form. The login form is submitted to Django view, password & username are matched, only if they match the website can be accessed, otherwise the site shows an error.

Similarly when a new user wants to register, he/she has to create a new account entering details including the username , full name, email, password & re-enter password.

Even at this time the details entered by the user are matched with database to check for similar data, an error pops up if data is not unique, username & email should be unique.

The user is redirected to the same page in case an invalid attempt is made. For example , if an invalid data is entered in the registration form , the user is redirected again to fresh registration form .

Similarly in the login page, the user is redirected back to the login page in case of invalid data or if the username and password don't match the data in the database.

User interactive environment consists of 3 main pages:

1. Homepage : Where user can login, register. If not logged in, or logout if logged in. Home page show data as per as if user is logged in or not.
2. Login : Where a user can login based on his/her credentials. Credentials are username and Password. In case the login details match to that of in the database the login is successful and user is redirected to the homepage of that particular user. Otherwise login attempt fails and the user is redirected back to the login page.
3. New User Registration : Where a new user can register, details are verified and validated.

Only then it is stored in the database and user is created. User is then redirected to the login page to login with those details.

Otherwise

```
urlpatterns = [  
    path('admin/', admin.site.urls), #admin page  
    path('login', login_view), #login page  
    path('accounts/register', register_view), #new user registration page  
    path('logout', logout_view), #logout page  
    path('', home_view) #homepage  
]
```

Fig. 3.1 URLs used in this project

#### TOOLS USED:

1. Django: Django is a Python-based free and open-source web framework that allows model-template-view architectural pattern and help us to create reliable & must faster websites.

```
(venv) C:\Users\SAHIL KAITH\PycharmProjects\FINAL\mysite>python -m django --version  
2.1.7
```

Fig. Django version used

2. Frontend Tools : HTML 5, CSS and BOOTSTRAP is used for interactive environment. A template based on BOOTSTRAP to show the static feature of Django.

3. SQLite as Database : SQLite is a simple SQL database engine. The code for SQLite is in the public domain and is open source and is thus free for use for any purpose, commercial or private. Being simple and easy to use and light to use make it the best choice.

4. Python : Django framework is based on python language so Python is used for scripting.

Jinja template language : Template language is used to creating functions within html code.  
For this, Jinja is used.

```
{% extends "base_generic.html" %}

{% block title %}{{ section.title }}{% endblock %}

{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

Fig. Syntax of Jinja template language.

### 3.2 Design Diagram :

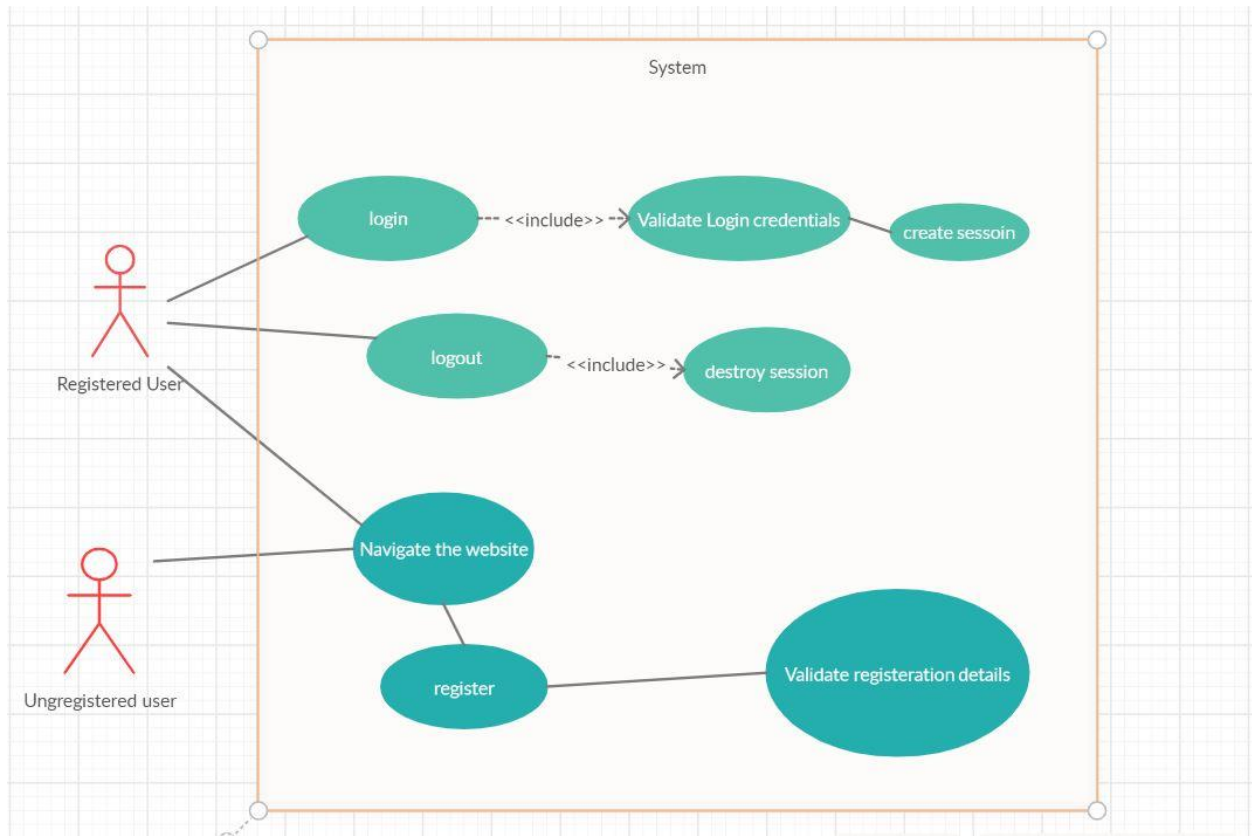


Fig.3.2 Use-case Diagram

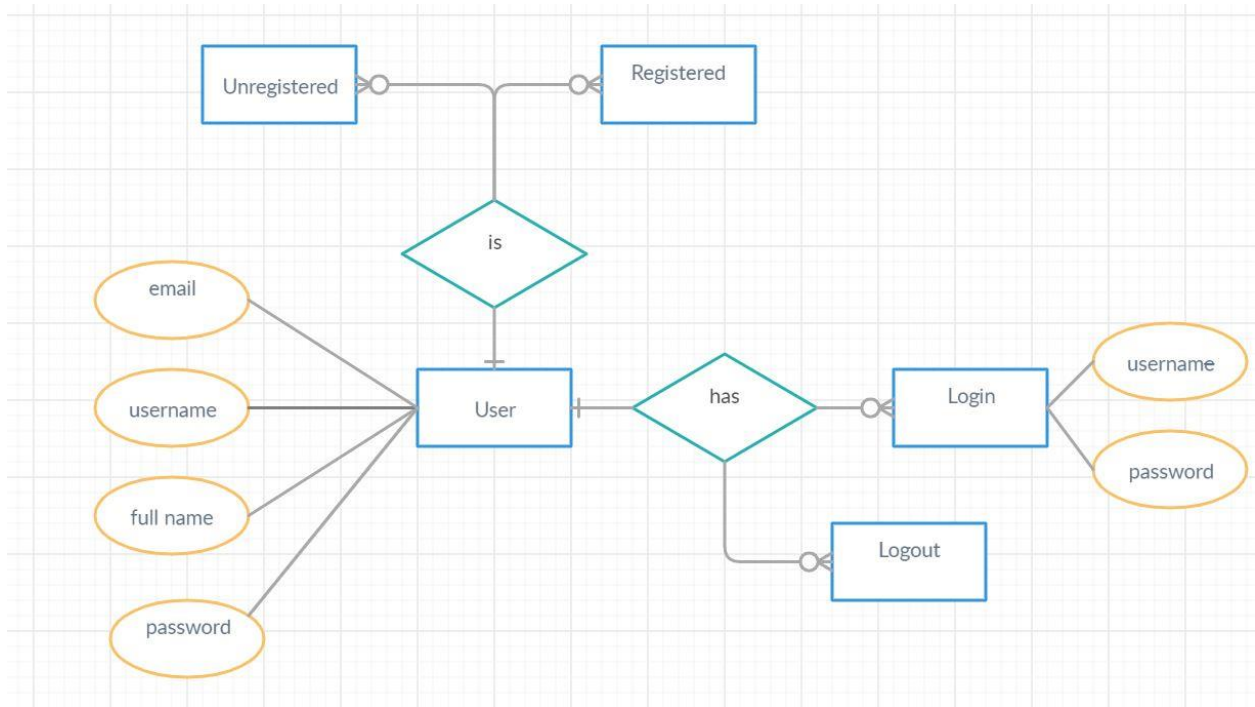


Fig 3.3 Entity Relationship diagram

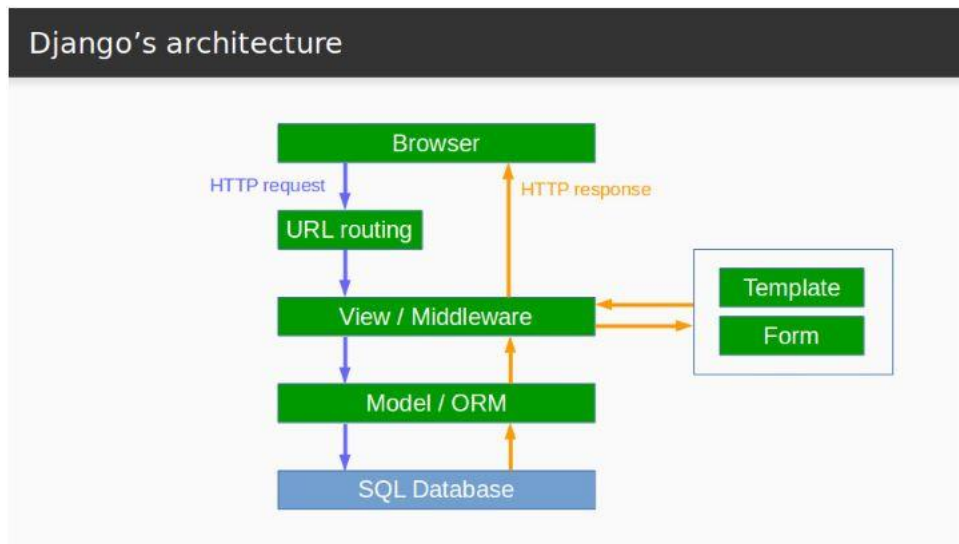


Fig. 3.4 Django Architecture



### **3.3 Functional And Non-Functional Requirement**

#### **3.3.1 Functional Requirement :** Funtional requirement for user management system are :

1. All users who access the website must be able to visit homepage regardless they are logged in or not. Option for login and new user registration must be given to every user that visit the website.
2. Users must be able to change their password and delete the account whenever they want to. User can also change their details provided it stays unique in the database.
3. User must be logged out completely when they want to.
4. User data must be protected at any cost consisting of login credentials and other user data.

#### **3.3.2 Non-Functional Requirement :**

1. User data must be protected at any cost.
2. The following application can be used on different operating systems, thus It offers portability.
3. The website is to be run on local server.

#### **3.3.3 Software Requirements :**

- Web Brower like Internet Explorer , Google Chrome , Microsoft Edge , Mozilla Firefox.
- IDE like Microsoft visual code , Sublime Text, Pycharm etc for toogling web pages.
- Virtual Environment.

## 3.4 Implementation Detail

In implementation of user management system we work on virtual environment to create a Django project.

**3.4.1 Setting up system for the project:** There are various things, which are needed before project can be started.

1. Install python.
2. Install pip & virtualenv
3. Set up virtual environment.
4. Install Django
5. Set up database
6. Text editor like Sublime text or Visual code

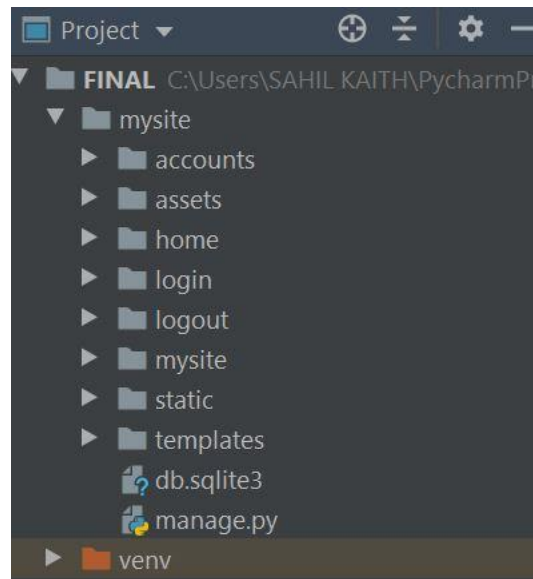


Fig.3.5 Root view of project

**Setting up database:** Database settings can be updated as per user requirements in setting.py in Project directory. By default MySQL lite is used for database purposes with Django Framework. User can also set up username & password for the database and other database services.

```

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

```

Fig. Database setting in settings.py

**Setting up static files :** These files include HTML, CSS , BOOTSPRAP and various other essential files that are included in templates

```

STATIC_URL = '/static/'
STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static')]
STATIC_ROOT = os.path.join(BASE_DIR, 'assets')

```

Fig. setting up folders for Static files.

**Setting up User-defined apps :** As per project requirements 3 user defined apps are included in setting.py.

```

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'login',
    'accounts',
    'logout',
]

```

Fig.3.6 Installed apps in setting.py

```
$ django-admin startproject mysite
```

Fig. Command to create new project in Django

```
$ python manage.py startapp polls
```

Fig. Command to create new

```
$ python manage.py runserver 8080
```

Fig. commandline to run local server.

Now the details of the app enlisted which help this project to get into shape , the accounts, login and home app work together to give the desired output.

### 3.4.2 Creating required apps within the project :

**Accounts:** This app deals with the registration of users and getting the data , validating it and storing it in database.

```
views.py x
1 from django.shortcuts import render, redirect
2 from django.contrib.auth.models import User
3 from django.contrib import messages
4 # Create your views here.
5 def register_view(request):
6     if request.method == 'POST':
7         username = request.POST['name']
8         email = request.POST['email']
9         password1 = request.POST['psw']
10        password2 = request.POST['psw-repeat']
11        if password1==password2:
12            if User.objects.filter(username=username).exists():
13                print('user name taken already')
14                messages.info(request, 'Username Taken')
15                return redirect("/accounts/register")
16            elif User.objects.filter(email=email).exists():
17                messages.info(request, 'email already taken')
18                return redirect("/accounts/register")
19            else:
20                user = User.objects.create_user(username=username, email=email, password=password1)
21                user.save();
22                print('user created')
23        else:
24            print('password do not match, user not created')
25            messages.info(request, 'password do not match')
26            return redirect("/accounts/register")
27        return redirect('/')
28    else:
29        return render(request, 'register.html', {})]
```

Fig.3.7 views.py of account app

This views.py in accounts app accounts for the validation of data that is entered in the registration form.

Certain libraries are used to perform the tasks.

1. Render : This is used to display the content of html file on the browser once the function is called of the views.py
2. Redirect : This help to redirect to the given address provided as the argument. It helps to redirect between pages efficiently.
3. User : User module in the Django.contrib helps to perform user related functions like saving the details of user.

Model.py help in achieving the structure of the data that is to be stored in database.

```
models.py
1 from django.db import models
2 class User(models.Model):
3     username = models.CharField(max_length=100)
4     password = models.CharField(max_length=50)
5     email = models.CharField(max_length=100)
6
```

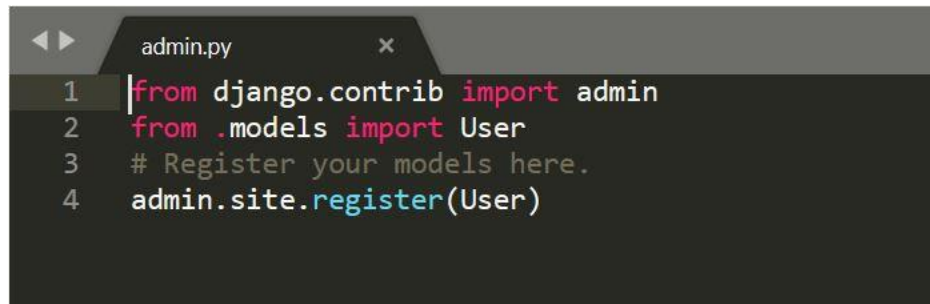
Fig. Model.py

Forms.py is necessary to let user enter a protected password because Django do not offer a specific text field for entering password.

```
forms.py
1 from django import forms
2 from .models import User
3 class UserForm(forms.ModelForm):
4     class Meta:
5         model = User
6         widgets = {
7             'password': forms.PasswordInput(),
8         }
```

Fig. form.py

Now, we have to register the model on admin.py. this is done with the following way.



```
admin.py
1 | from django.contrib import admin
2 | from .models import User
3 | # Register your models here.
4 | admin.site.register(User)
```

Fig Admin.py

**Login:** This app is created to perform the login operation and displaying the login page on the website. This is made using a pre-made template who's data is stored in static folder.

Django allows us to access the database and help retrieving the data and perform operation on them.

Here the login credentials are fetched from database or rather the login credentials entered by the user are matched with the credentials in the database. Auth library is used for this purpose.

The views.py of the login app helps to visualize the login page of the website and also the authentication of the user.

It uses the concept of POST and GET method beautifully.

```
views.py
1 from django.shortcuts import render, redirect
2 from django.contrib import auth
3
4 # Create your views here.
5 def login_view(request):
6     if request.method=='POST':
7         username = request.POST['username']
8         password = request.POST['pass']
9         user = auth.authenticate(username=username, password=password)
10        if user is not None:
11            auth.login(request, user)
12            return redirect('/')
13        else:
14            return redirect("/login")
15
16
17    else:
18        return render(request, 'login.html', {})
19
```

Fig3.8 views.py of Login app.

## POST AND GET METHOD :

GET and POST are the two HTTP methods which are used when we deal with forms.

POST method is used to return Django's login form, in which the browser collects the form data, encodes it for sending, sends it to the server, and then receives back its response.

Any request that will be used to change the state of the system - for example, a request that makes changes in the database, provides data into database - should always use POST method. GET is used only for requests that do not bring changes in the website.

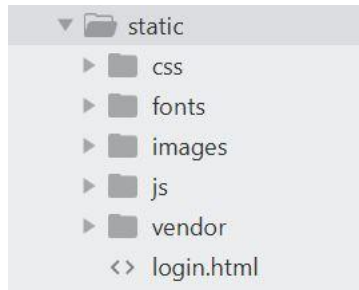
GET should not be used for a password form, because the password will then appear in the URL, and in browser history and server logs, all in plain text. Neither it is suitable for large amount of data, or for binary data, such as an image, videos.

A Web application that uses GET requests for admin forms is also a security risk: it can be easy for an attacker to mimic a form's request to gain access to sensitive parts of the system. POST, along with protections like Django's CSRF protection offers more security.

On the other hand, GET is more suitable for things like a web search form, because the URLs that represent a GET request can easily help to redirect and search.

**STATIC Directory :** The login page has all its data stored in static folder.





The static files are loaded into the login.html like following :

```
login.html
1  {% load static %}
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5      <title>LOGIN</title>
```

For every location of files in the static folder the Jinja format is used.

```
<!-->
<link rel="icon" type="image/png" href="{% static 'images/icons/favicon.ico' %}"/>
<!-->
<link rel="stylesheet" type="text/css" href="{% static 'vendor/bootstrap/css/bootstrap.min.css' %}">
<!-->
<link rel="stylesheet" type="text/css" href="{% static 'fonts/font-awesome-4.7.0/css/
font-awesome.min.css' %}">
<!-->
<link rel="stylesheet" type="text/css" href="{% static 'fonts/Linearicons-Free-v1.0.0/
icon-font.min.css' %}">
<!-->
<link rel="stylesheet" type="text/css" href="{% static 'vendor/animate/animate.css' %}">
<!-->
<link rel="stylesheet" type="text/css" href="{% static 'vendor/css-hamburgers/hamburgers.min.css' %}">
<!-->
<link rel="stylesheet" type="text/css" href="{% static 'vendor/animation/css/animation.min.css' %}">
<!-->
<link rel="stylesheet" type="text/css" href="{% static 'vendor/select2/select2.min.css' %}">
<!-->
<link rel="stylesheet" type="text/css" href="{% static 'vendor/daterangepicker/daterangepicker.css' %}">
<!-->
<link rel="stylesheet" type="text/css" href="{% static 'css/util.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'css/main.css' %}">
```

Fig Static files are accessed by using special template language in href tag.

**Logout:** This app deals with the logout feature. User can end the session by clicking on logout and then they are redirected to the home page. User can then login again with the same account or different account. User also has the option to register a new account from homepage.

The views.py of the logout app deals with the logout request.  
For this we import auth module from Django.contrib.

Django user authentication system consists of:

- Users
- Permissions: Binary (yes/no) flags depicting whether a user can perform a certain task.
- A configurable password hashing system to save passwords.
- Forms and view tools for logging in users, or restricting content
- A backend system

The authentication system in Django aims to be very basic .  
For features which Django do not provide are implemented in third-party packages:

```
views.py x
1 from django.shortcuts import render, redirect
2 from django.contrib.auth.models import auth
3 # Create your views here.
4 def logout_view(request):
5     auth.logout(request)
6     return redirect('/')
```

Fig. 3.9 views.py of logout

The views.py consist of a single method called `logout_view` which helps user get logout of the website.

The user is then redirected to the homepage of the website for further login or new user registration.

**Home:** This app deals with homepage of the website. From here a user have the option to login , logout if already logged in. User can also create a new accout. The links to every page is maintained.

```
views.py x
1 from django.shortcuts import render
2
3 # Create your views here.
4 def home_view(request):
5     return render(request, 'home.html')
```

Fig3.10 views.py of homepage

The views.py of the homepage helps us to render out the home.html that is the html page of our homepage.

```
<body>

<ul>
  <li><a class="active" href="/">Home</a></li>

  <li><a href="/accounts/register">New User Registration</a></li>
  {% if user.is_authenticated %}
  <li><a href="/logout">Logout</a></li>
  {%else%}
  <li><a href="/login">Login</a></li>
  {%endif%}
</ul>
  {% if user.is_authenticated %}
  <h3>Hello, {{user.username}}</h3>
  {% else %}
  <h3>Click on login to access</h3>
  {%endif%}
</body>
</html>
```

Fig.3.11 Body of home.html

In home.html simple if condition is used to check if the user is logged in or not. This is done by is\_authenticated attribute of user module.

Using class models.User , is\_authenticated attribute is used.

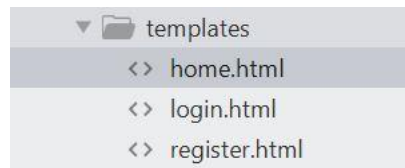
```
home.html x
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5 ul {
6     list-style-type: none;
7     margin: 0;
8     padding: 0;
9     overflow: hidden;
10    background-color: #333;
11 }
12
13 li {
14     float: left;
15 }
16
17 li a {
18     display: block;
19     color: white;
20     text-align: center;
21     padding: 14px 16px;
22     text-decoration: none;
23 }
24
25 li a:hover {
26     background-color: #111;
27 }
28 </style>
29 </head>
30 </html>
```

Fig. 3.12 head of home.html

Simple style tag is used to add appearance to the home page of the website.

The style is applied to all the links allowed for the user.

**Templates:** separate folder is maintained for the templates which are to be used to display different pages of website. These are accessed by the views.py of their respective app.



For our app we have three pages for our website , so 3 templates are used for each function.

**URL.PY :** This python file manages all url redirecting and url addresses. The view.py of every app is mapped to this file.

```
from django.contrib import admin
from django.urls import path
from login.views import login_view
from accounts.views import register_view
from home.views import home_view
from logout.views import logout_view
urlpatterns = [
    path('admin/', admin.site.urls), #admin page
    path('login', login_view), #login page
    path('accounts/register', register_view), #new user registration page
    path('logout', logout_view), #logout page
    path('', home_view) #homepage
]
```

Fig. Url.py in project directory

### **3.5 Implementation Issues :** There are various implementation issues in this which include:

1. The virtual environment should be always running while working on the project only the the website can be run on the local server.
2. Database should be always connected to the framework so that new user can always be registered and the login attempts are recorded as well as verified.
3. Even when new user are entering their details, the details are matched with the database to check for similar data. So uninterrupted connection of database is must.
4. Web browsers should support all the template design and Django framework. The template files should be loaded by the browser. It should support all technologies.
5. Url mapping , POST & GET request are to be carefully implemented.
6. The csrf token has to be used for the data for security purposes , otherwise the data can't be transferred from the forms

## **CHAPTER 4**

### **PERFORMANCE ANALYSIS**

#### **4.1 Agile Methodology**

Agile Methodology is basically a process in which we do constant iterations of production and testing phase whole time during the software development life cycle and contains more advantages over waterfall model. If we talk about waterfall model then it initially required to be fully designed then it is made forward for doing testing but in agile model we could produce a few of product commit it and then do parallel testing and finally verify it. It is pretty simple to make alters in the program and deployment of product is also quick. I am following agile method in this user management system.





## 4.2 Analytics

- In this project, we are to monitor what is going on in the backend of server. How data is being processed and how GET & POST commands are working. For this we use cmd in terminal to print out the details of flow of data for better understanding
- Also in Google Chrome we use Inspect element and console features to tweak the scripts and template. This give us better understand of how the program is working.
- Refresh command on browsers let user see quick changes in their templates.

```
Terminal: Local x +
Microsoft Windows [Version 10.0.18363.592]
(c) 2019 Microsoft Corporation. All rights reserved.

(venv) C:\Users\SAHIL KAITH\PycharmProjects\FINAL>
```

Fig. Terminal in Pycharm is used to run commands for the project

### 4.3 OUTPUTS :

```
Terminal: Local x +
(venv) C:\Users\SAHIL KAITH\PycharmProjects\FINAL>cd mysite

(venv) C:\Users\SAHIL KAITH\PycharmProjects\FINAL\mysite>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
June 23, 2020 - 11:06:52
Django version 2.1.7, using settings 'mysite.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Fig 4.3.1 The command to start server and the url of homepage.

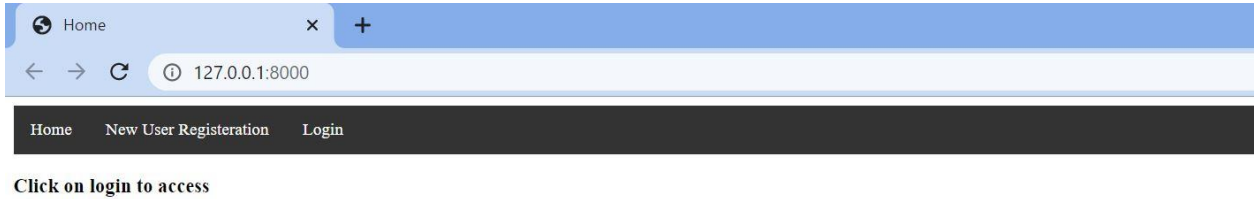


Fig 4.3.2 The homepage where user has option to login & create new account.

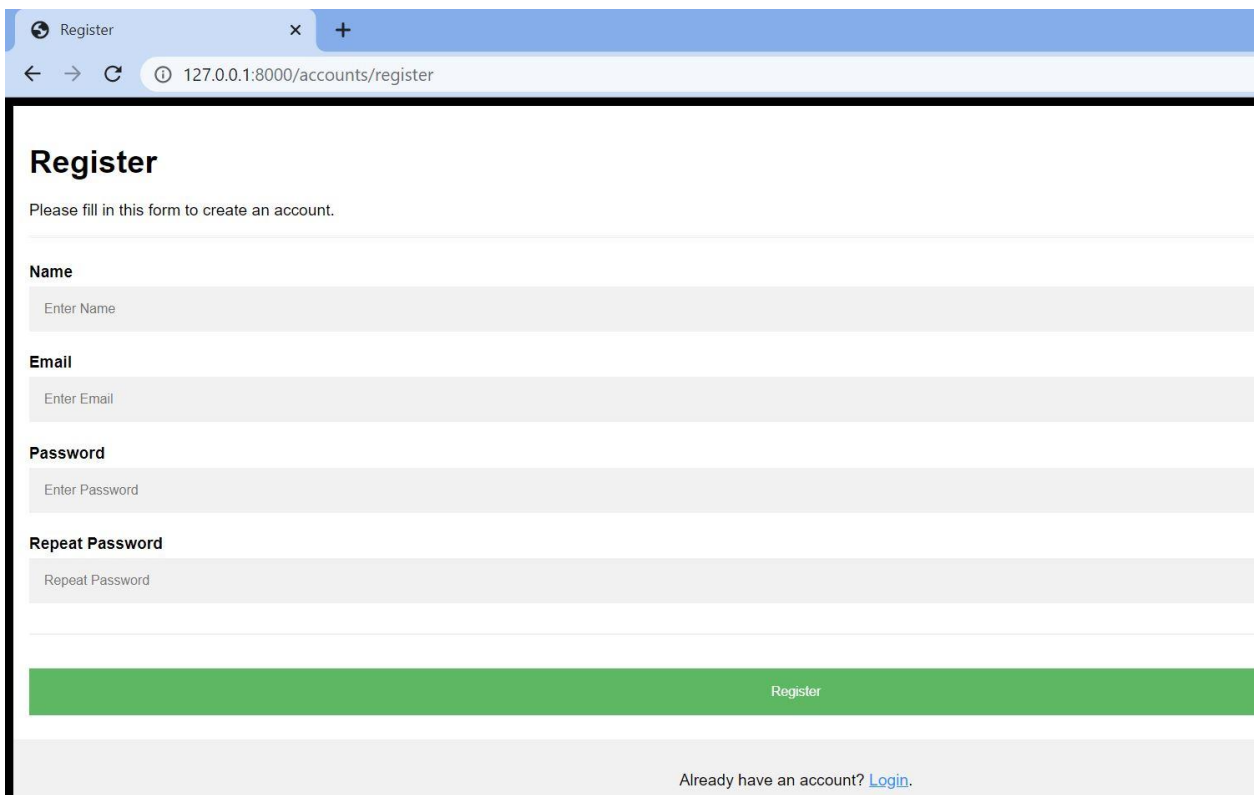


Fig. 4.3.3 The registration form incase user wishes to create a new account.

User can also redirect back to the login page if he/she already has an account.

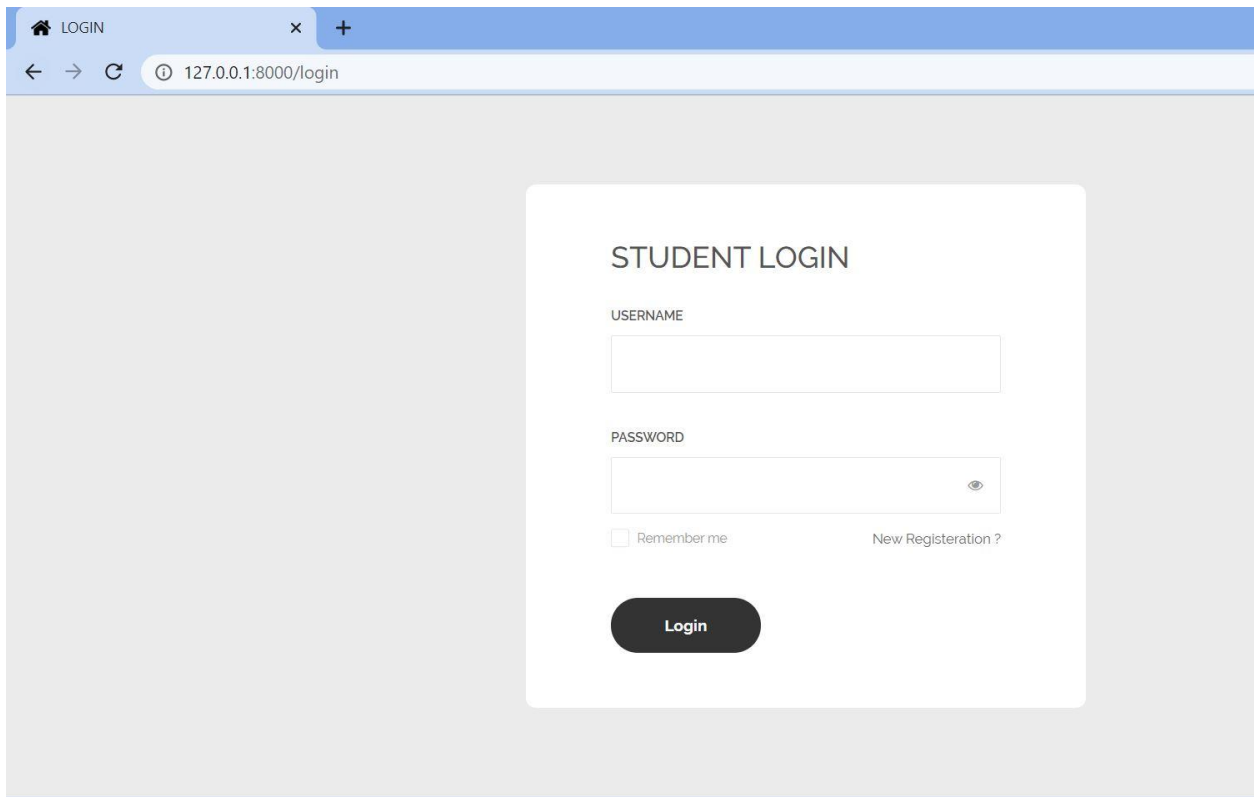


Fig 4.3.4 Login page where user can login with login credentials.

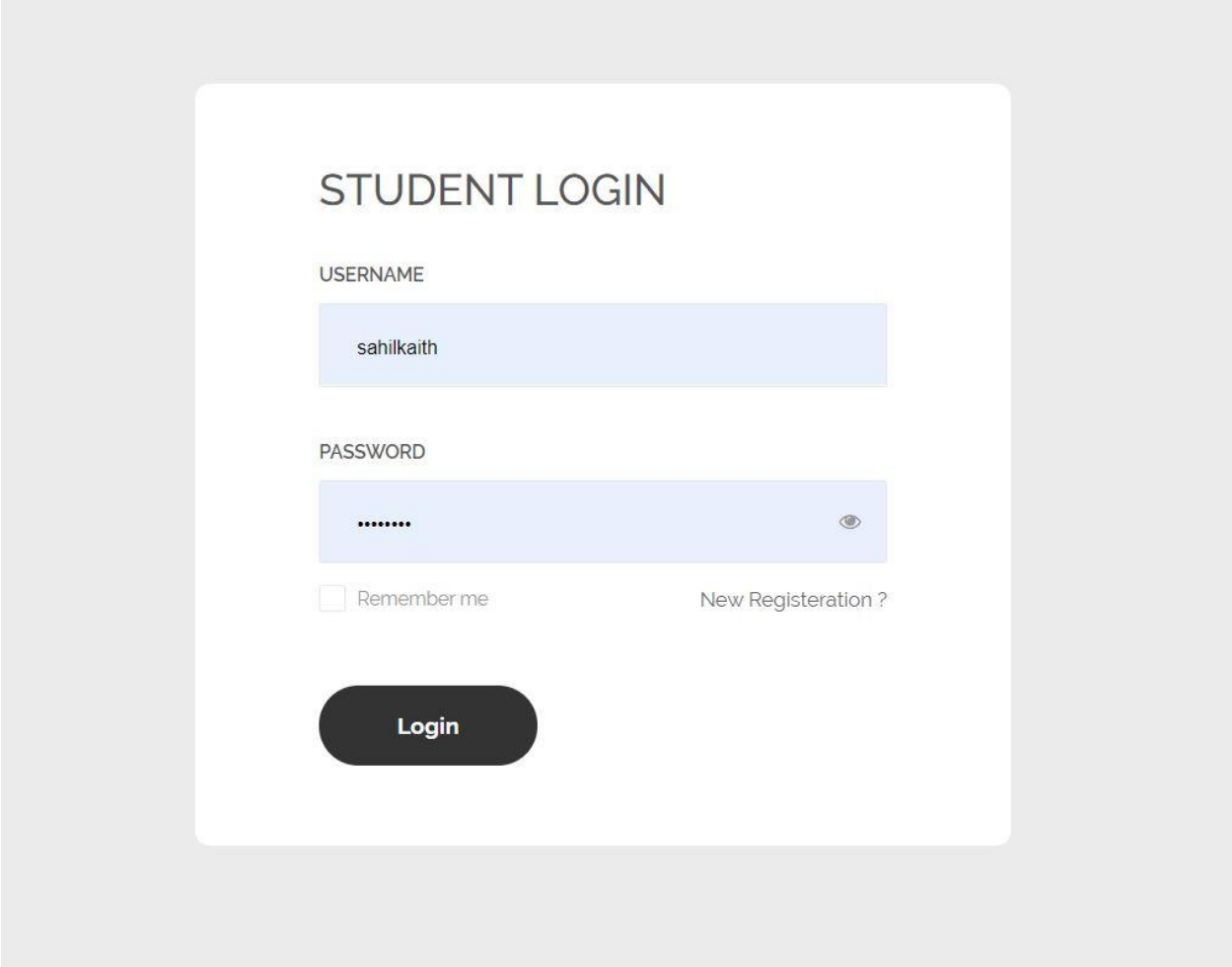


Fig. 4.3.5 Me signing in with my created account.

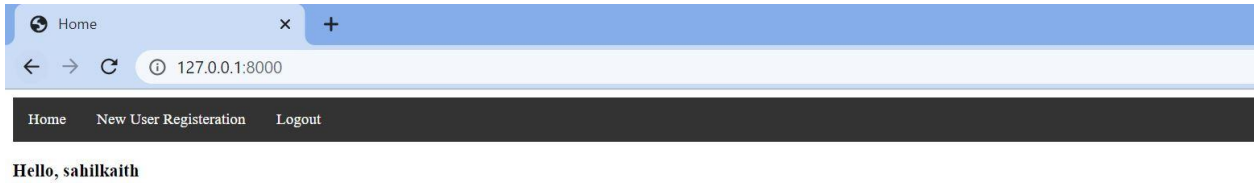


Fig. 4.3.6 The home page content changes as per the user who is signed in. We can create any number of users.

User can logout when logged in. The login tab is replaced by logout.

## **CHAPTER 5**

### **CONCLUSIONS**

#### **5.1 CONCLUSION**

So, this is the last chapter of report where there is whole conclusion of the report. We hereby conclude the importance of user management system in every website. How necessary they are for better interaction with the users.

We learnt the importance of confidentiality of the user data. How website access should be restricted to the members who have registered.

The marketing value of adding more users to the website and better interaction of user with the website.

Every user has his/her own requirement, so content of the website should be according to the user who is logged in.

User management is such an integral part of a website which should not be neglected. No website can flourish without user connectivity.

Employees are an important part of every organization, similarly the users are an important part of every web application.

The scope of big companies like Facebook, Google, Twitter, YouTube can not be imagined without billions of users who are logged into it.

#### **5.2 FUTURE SCOPE**

If throwing some light on the future of this program, so this project has great scope in future. In future we will include many features to this program.

We have countless amendments to make on this project from security & encryption of data to the frontend work.

- We can provide encryption algorithms to save user data.
- Password combinations can be made more secure by using combination of alpha numeric & symbols.
- The front end interface of user to login and create new account can be made more appealing and as the same time more secure and robust
- Email validation can be the most important step we need to make on this project. Verifying email is major step every website make sure to do whenever a new user sign up. This helps to ensure user is valid and there are less security thefts.
- The form validation can be made more secure by providing column for security question , otp and pin.
- The user can be provided with option to change his/her details including username and password.

So there are some factors on which in future we can think on applying and making it more reliable.



## REFERENCES

1. <https://realpython.com/django-setup/#create-an-app>
2. <https://www.codingforentrepreneurs.com/blog/install-python-django-on-windows/>
3. <https://www.freecodecamp.org/>
4. <https://docs.djangoproject.com/en/3.0/>
5. <https://stackoverflow.com/>
6. <https://www.youtube.com/watch?v=OTmQOjsl0eg>
7. <https://www.w3schools.com/>
8. <https://docs.djangoproject.com/en/3.0/howto/static-files/>
9. <https://docs.djangoproject.com/en/3.0/topics/auth/>
10. <https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/>

## sahil report2

### ORIGINALITY REPORT

<b>18%</b>	<b>14%</b>	<b>1%</b>	<b>6%</b>
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

### PRIMARY SOURCES

<b>1</b>	<b>buildmedia.readthedocs.org</b> Internet Source	<b>9%</b>
<b>2</b>	<b>studylibr.com</b> Internet Source	<b>2%</b>
<b>3</b>	<b>docs.djangoproject.com</b> Internet Source	<b>1%</b>
<b>4</b>	<b>Submitted to University of Wales, Bangor</b> Student Paper	<b>1%</b>
<b>5</b>	<b>Daniel Rubio. "Beginning Django", Springer Science and Business Media LLC, 2017</b> Publication	<b>1%</b>
<b>6</b>	<b>masteringdjango.com</b> Internet Source	<b>1%</b>
<b>7</b>	<b>Submitted to University of Durham</b> Student Paper	<b>1%</b>
<b>8</b>	<b>kepofucypi.ga</b> Internet Source	<b>&lt;1%</b>
<b>9</b>	<b>viaalexito.com</b>	

	Internet Source	<1 %
10	Submitted to University of Liverpool Student Paper	<1 %
11	Submitted to De Montfort University Student Paper	<1 %
12	<a href="http://linda-project.eu">linda-project.eu</a> Internet Source	<1 %
13	Submitted to INTI International University Student Paper	<1 %
14	Submitted to Kampala International University Student Paper	<1 %
15	<a href="http://ia-linncounty2.civicplus.com">ia-linncounty2.civicplus.com</a> Internet Source	<1 %
16	Submitted to University of Portsmouth Student Paper	<1 %
17	Submitted to University of Southampton Student Paper	<1 %
18	<a href="http://elaunchr.ominfosolutions.in">elaunchr.ominfosolutions.in</a> Internet Source	<1 %
19	Submitted to Segi University College Student Paper	<1 %
20	Submitted to Visvesvaraya Technological	<1 %

# University, Belagavi

Student Paper

---

---

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		

# JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

## PLAGIARISM VERIFICATION REPORT

Date: .....15-07-2020.....

Type of Document (Tick):  PhD Thesis  M.Tech Dissertation/ Report  B.Tech Project Report  Paper

Name: SAHIL KAITH Department: C.S.E. Enrolment No 161247

Contact No. 7018073720 E-mail. sahil.kaith555@gmail.com

Name of the Supervisor: Dr. Hari Singh

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): USER MANAGEMENT SYSTEM

### UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

#### Complete Thesis/Report Pages Detail:

- Total No. of Pages = 52
- Total No. of Preliminary pages = 10
- Total No. of pages accommodate bibliography/references = 1

(Signature of Student)

### FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....<sup>18</sup>.....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

### FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"><li>• All Preliminary Pages</li><li>• Bibliography/Images/Quotes</li><li>• 14 Words String</li></ul>		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by  
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at [plagcheck.juit@gmail.com](mailto:plagcheck.juit@gmail.com)