



NITTE
(Deemed to be University)

**NMAM INSTITUTE
OF TECHNOLOGY**

Nitte (DU) established under Section 3 of UGC Act 1956 | Accredited with 'A+' Grade by NAAC

Department of Computer Science and Engineering

Report on Mini Project

Social Media Website

Course Code : 20CSE41

Course Name : Web Programming

Semester: V SEM

Section: C

Submitted To:

Dr Sarika Hegde
Associate Professor
Department of Computer Science
and Engineering

Submitted By:

Prathiksha Kini
4NM20CS139

Shanelle D'Mello
4NM20CS161

Date of submission: 17-12-2022

Signature of Course Instructor

ABSTRACT

Social networks constitute the greatest global information platform on the internet today. They have become an indispensable part of our daily lives, As people spend more time socializing on the internet.

The web project developed by Prathiksha Kini and Shanelle D'Mello simulates a social media website called "Truffle" where the users can create, edit and update their profiles with a strong authentication system provided inbuilt by the Django framework. The logged in users can add, edit, download and delete posts, search users, like and comment on the post as well as follow/unfollow other users. The website hosts a user suggestion section where random users are displayed on refreshing the page.

This project report will introduce how to build a social media web application system using the Django framework. Django is an open-source web application framework written in python. This social media web application system built using Django has five major components with different functionalities that will be introduced later. We will introduce various features of the Django framework and SQLite3 RDBMS in the later section. In the end, snapshots are attached to demonstrate UI.

TABLE OF CONTENTS

TitlePage	1
Abstract.....	2
TableofContents	3
Introduction	4
ProblemStatement.....	4
Objectives	5
Hardware/Software Requirement.....	6
Methodology	7
ImplementationDetails	10
Results	13
Conclusion andFutureScope	16
References	17

INTRODUCTION

The project aims to create a desirable social media application for users with a suitable UX design and proper backend management.

- The project uses the Django web framework (python framework) to implement the social media application.
- The project uses SQLite3 software library for a relational database management system in the backend to store, retrieve, and perform necessary backend operations.
- HTML, Bootstrap, CSS, SCSS and Django Template Language implement the front-end to customize the user interface.

PROBLEM STATEMENT

Truffle is a Django based social media website where users can utilize the features exactly as on other straightforward social media websites. The goal of the project was to demonstrate how the most popular web development languages might be utilized to create a straightforward social media website.

Features:

As mentioned above, this application system has five vital components/features: User registration/authentication, User posts, Search users, Like and Comment on user posts and Follow/Unfollow users.

- **User registration/Authentication:** Any application will include this primary feature to register users in their application. To access all the other features, the user must register into the application. We collect users fundamental data such as email, username, first name, last name, password, and store it in the database. Moreover the users can change their password for security purposes.
- **Profile modification:** This is an extended feature of user registration. Here, users can create and modify their profiles. Users can change their profile picture, email address, and their usernames. The altered data reflects in the database system and the front-end of the application.
- **User posts:** Once the user has registered and set up a profile, they can post photos and modify them accordingly. Users can like and comment and download the posts.
- **Follow/Unfollow users:** The logged in user can follow/unfollow users from the user suggestion section.
- **Search users:** Users can search other users by entering the username in the search bar provided in the home page. A list of all users matching the username typed will appear as a search result.

Objective:

Presenting a straightforward yet distinctive website that suits the demands of a user like any other social media websites. The project created is a social media web application where visitors may register and utilize the features exactly as on other straightforward websites. The system will be built using the Django web framework and the Bootstrap framework for the frontend design.

HARDWARE / SOFTWARE Requirements

Hardware Requirements:

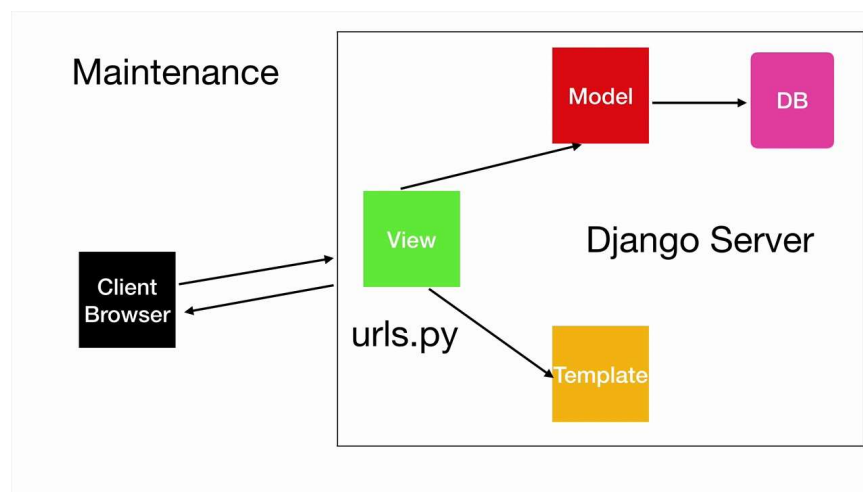
Django framework doesn't have many system requirements. Since it's primarily written in python, the system requirements must satisfy to install Python language. Similarly, SQLite3 can also run on any modern system. Hardware requirements for this project are primarily concerned with the system hardware requirements (PC). We recommend Intel Core i3 processor or above for the development phase with a minimum 4GB PC Ram and HDD or SSD hard disk. Server end hardware requirements aren't significant for this project because we can host this project on the cloud with various services.

Software Requirements:

This social media application application uses python language for development with SQLite database management system. We have used the Django framework version 4.1.3 and hence we have to use python version 3.6 or above and SQLite version 3.8 or above. The Integrated Development Environment used for this project is Visual Studio Code.

METHODOLOGY

The project will be developed using an iterative and incremental approach. The first step will be to define the requirements and design the system architecture. This will involve identifying the various components of the system and their interactions, as well as defining the user flows and functionalities. Once the requirements and design are finalised, the implementation phase will begin. This will involve the development of the various components of the system, including the user interface, database, and backend logic. The system will be tested and refined throughout the development process to ensure that it meets the requirements and provides a smooth and efficient user experience.



Django is an open-source web application framework written in python. It is a high-level framework that encourages rapid development and clean, pragmatic design. Django consists of three major parts: Model, View, and Template.

Model: Model is a single, definitive data source that contains the essential field and behavior of the data. Python classes implement models. Usually, one model is one table in the database. Each python object in the model represents a field of a table in the database. Django provides a set of automatically generated database application programming interfaces (APIs) for the convenience of users.

View: A view is a short form of the view file. It is a file containing a Python function that takes web requests and returns web responses. A response can be HTML content or XML documents or a “404 error” and so on. The logic inside the view function can be arbitrary as long as it returns the desired response. To link the view function with a particular URL, we need to use a structure called URLconf that maps URLs to view functions.

Template: Django’s template is a simple text file that can generate a text-based format like HTML and XML. The template contains variables and tags. Variables will be replaced by the result when the template is evaluated. Tags control the logic of the template. We also can modify the variables by using filters. For example, a lowercase filter can convert the variable from uppercase into lowercase. Django templates allow the developers to implement the front-end logic of the application.

Step 1 : The UI for the project was designed using HTML, CSS, SCSS, Bootstrap and Javascript.

Step 2: Once the frontend was designed a schema was created for different models:

```
class Profile(models.Model):
    # user = models.ForeignKey(User, on_delete=models.CASCADE)
    user=models.OneToOneField(User,null=True,on_delete=models.CASCADE)
    description = models.TextField(blank=True)
    fname = models.TextField(blank=True)
    lname = models.TextField(blank=True)
    username=models.TextField(blank=True)
    profileimg = models.ImageField(upload_to='profile_images', default='blank-profile-picture.png')

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

    def get_absolute_url(self):
        return reverse('home')
```



```

class FollowersCount(models.Model):
    follower = models.CharField(max_length=100)
    user = models.CharField(max_length=100)

    def __str__(self):
        return self.user

class Post(models.Model):
    title=models.CharField(max_length=255)
    image=models.ImageField(null=True,blank=True,upload_to="images/")
    title_tag=models.CharField(max_length=255,default="")
    author=models.ForeignKey(Profile,on_delete=models.CASCADE)
    caption=RichTextField(blank=True,null=True)
    post_date=models.DateField(auto_now_add=True)
    location=models.CharField(max_length=255,default="")
    no_of_likes=models.IntegerField(default=0)

```

```

class Comment(models.Model):
    post=models.ForeignKey(Post,related_name="comments",on_delete=models.CASCADE)
    name=models.CharField(max_length=255)
    body=models.TextField()
    date_added=models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return '%s - %s' % (self.post.title,self.name)

    def get_absolute_url(self):
        return reverse('home')

class LikePost(models.Model):
    post_id=models.CharField(max_length=500)
    username=models.CharField(max_length=100)

    def __str__(self):
        return self.username

```

Step 3: Views are written to write the underlying logic that needs to be displayed on the frontend in views.py file and routes are configured in urls.py file.

IMPLEMENTATION

Installation Steps:

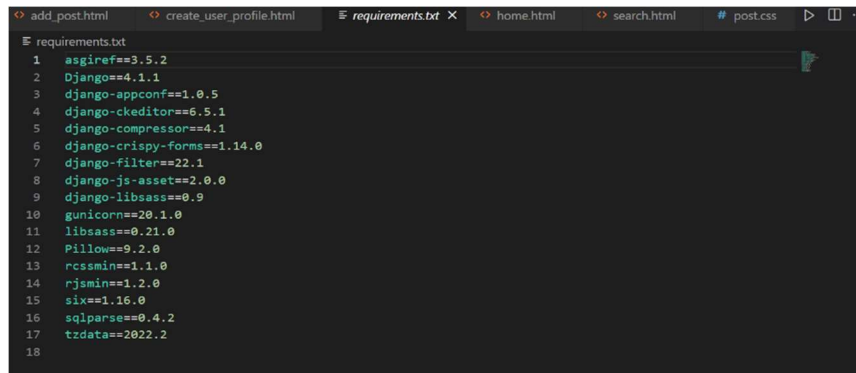
Step 1: Open VS Code or any other editor of your choice and Install virtualenv using the command **pip install virtualenv** in the terminal.

Step 2: Run the below 2 commands:

virtualenv env

env/scripts/activate

Step 3 : Install all the requirements by running **pip install -r requirements.txt**

A screenshot of a code editor window with a dark theme. The editor shows a file named 'requirements.txt' with 18 lines of code. The code lists various Python packages and their versions, including asgiref, Django, django-appconf, django-ckeditor, django-compressor, django-crispy-forms, django-filter, django-js-asset, django-libsass, gunicorn, libsass, Pillow, rcssmin, rjsmin, six, sqlparse, and tzdata.

```
1 asgiref==3.5.2
2 Django==4.1.1
3 django-appconf==1.0.5
4 django-ckeditor==6.5.1
5 django-compressor==4.1
6 django-crispy-forms==1.14.0
7 django-filter==22.1
8 django-js-asset==2.0.0
9 django-libsass==0.9
10 gunicorn==20.1.0
11 libsass==0.21.0
12 Pillow==9.2.0
13 rcssmin==1.1.0
14 rjsmin==1.2.0
15 six==1.16.0
16 sqlparse==0.4.2
17 tzdata==2022.2
18
```

Step 4: Create database table by applying migrations

python manage.py makemigrations
python manage.py migrate

Step 5: Finally run the command **python manage.py runserver**

A development server will be started run the website in any browser and ensure proper internet connection.

The Python and Django built in Functions used for the project:

- Python's in built random function was used to shuffle the users randomly in user suggestions section.
- Len function was used to find the number of posts posted by user, number of followers etc.

Python query sets and SQL:

This section covers most of the Django query sets and raw SQL that have been used in the blog application. Query sets are the list of objects that have been created using the models. We can perform operations such as add, delete, retrieval, and many more. It's implemented in python and can interact with database tables.

- **Filter method:** filter() method is used when we need to perform queries with certain conditions. We have used the filter method to filter the blogs based on category.
- **All method:** The simplest way to retrieve an object (tuple) from the database table is to call all() method when using query sets.
- **__icontains method:** The simplest way to search an object (tuple) from the database table is to call __icontains() method when using query sets.

Generic Views: Writing web applications can be monotonous, because we repeat certain patterns again and again. Django tries to take away some of that monotony at the model and template layers, but web developers also experience this boredom at the view level.

Views used:

- **ListView :** List View refers to a view (logic) to display multiple instances of a table in the database.
- **DetailView:** Detail View refers to a view (logic) to display one instances of a table in the database.
- **CreateView :** Create View refers to a view (logic) to create an instance of a table in the database.

- **PasswordChangeView:** Django provides authentication and authorization. For Changing Password you need to get authenticated first. In your urls.py, we need to import PasswordChangeView from Django Auth. By default, Django PasswordChangeView will render template base/change_password.
- **UpdateView:** Update View refers to a view (logic) to update a particular instance of a table from the database with some extra details.
- **DeleteView:** Delete View refers to a view (logic) to delete a particular instance of a table from the database.

Models are created for each table with its attributes specified. The operations are then done with inbuilt methods in Django.

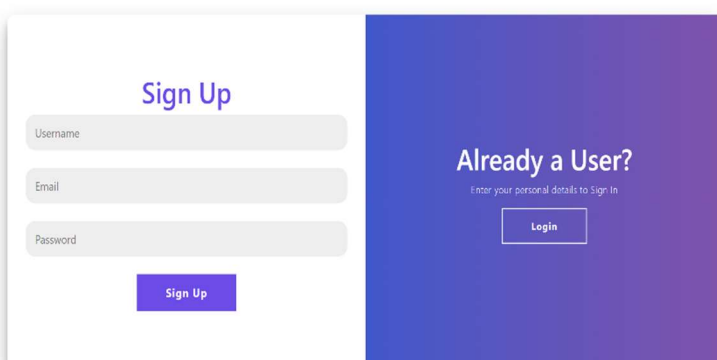
CRUD in Django

- Inserting is done by the save() method
- Deleting with delete() method
- Updating with is done by changing the fields for the fetched object , again with save() method

RESULTS AND DISCUSSIONS

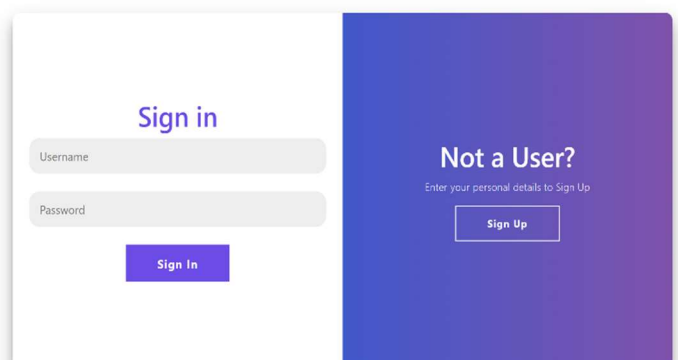
Snapshots of the project:

Signup Page



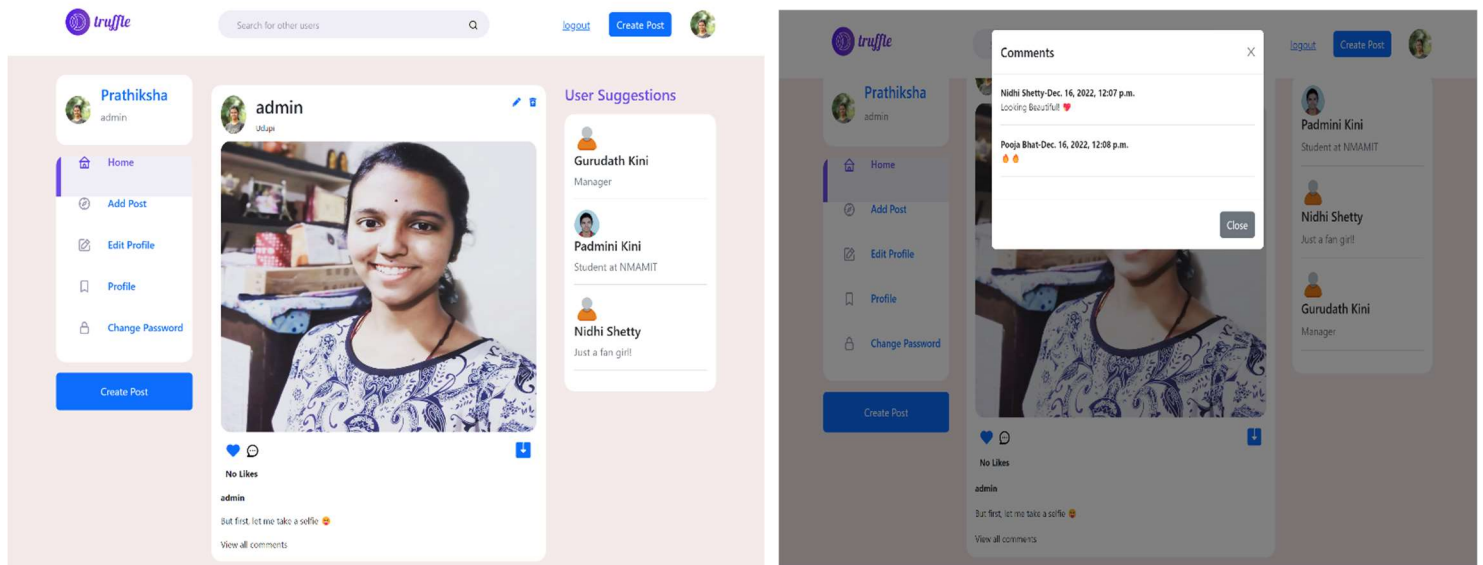
The Signup Page features a white background on the left and a blue-to-purple gradient on the right. On the white side, there is a 'Sign Up' heading, three input fields for 'Username', 'Email', and 'Password', and a purple 'Sign Up' button. On the gradient side, there is a link 'Already a User?' with the text 'Enter your personal details to Sign In' and a white 'Login' button.

Login Page

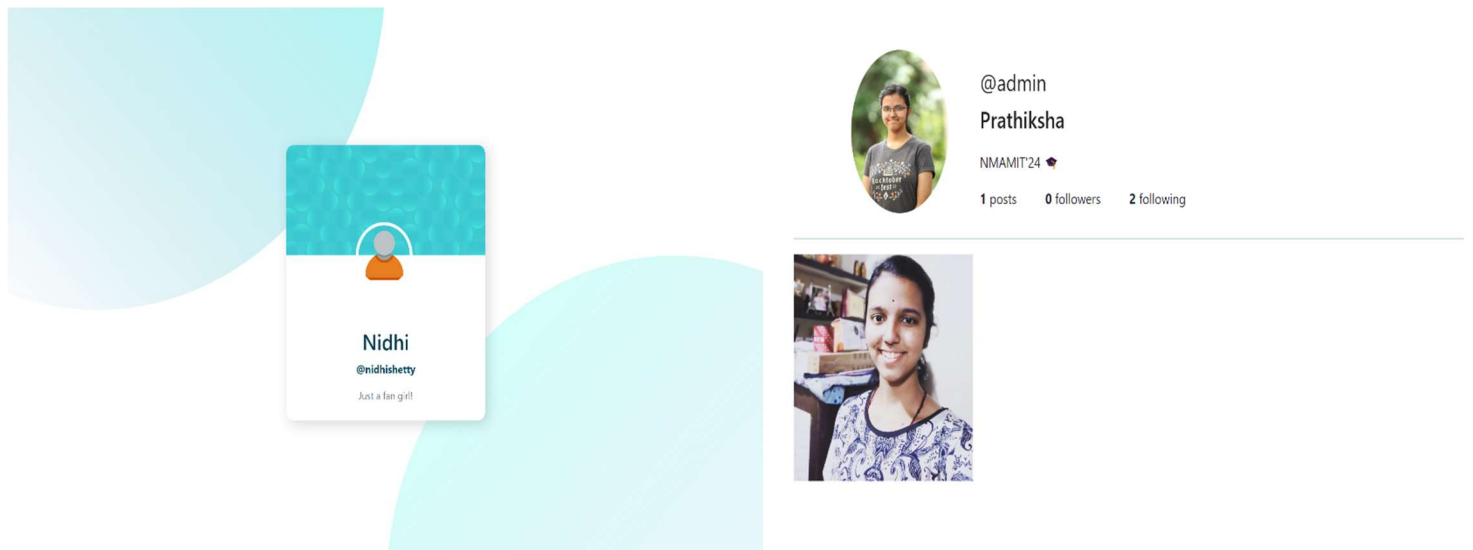


The Login Page features a white background on the left and a blue-to-purple gradient on the right. On the white side, there is a 'Sign in' heading, two input fields for 'Username' and 'Password', and a purple 'Sign In' button. On the gradient side, there is a link 'Not a User?' with the text 'Enter your personal details to Sign Up' and a white 'Sign Up' button.

Home Page and Comment Display Modal



Search user results page and User Profile Page



Django Form-Validation

Django provides built-in methods to validate form data automatically. Django forms submit only if it contains CSRF tokens. It uses a clean and easy approach to validate data.

The **is_valid()** method is used to perform validation for each field of the form, it is defined in Django Form class. It returns True if data is valid and place all data into a `cleaned_data` attribute.

CONCLUSION AND FUTURE SCOPE

As the goal was set most of the implementations and requirements have been passed and test cases have been solved. Some problems for developer like error correction did become a hassle as we are new to this subject, but at the end a fruitful website was created together.

Future Scope can include integrating our application with a real time chat application that could serve the demand of the users on a successful social media web application. Working on the backend can be time-consuming and yet can be simplified when working with the Django framework. SQLite3 provides a robust and flexible Database Management System that is very suited for the scalability of products.

Extensions or development of the database tables so that a better database can be provided. No software is perfect, good maintenance and update to the trend is what will make any software shine.

SQLite3 database works at its best, backed by the authentication and security features of Django.

REFERENCES

- Django documentation: <https://www.djangoproject.com>
- Python documentation: <https://docs.python.org>
- Stack Overflow Solutions
- Sqlite3 documentaion <https://www.sqlite.org/docs.html>

