MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Mini Project Report

of

Web Programming Lab (CSE 3243)

Social Media Platform

SUBMITTED BY

AYANIKA PAUL - 220905128 – 22 - CSE D
ISHIKA JAISWAL – 220905140 –24 - CSE D
TUSHAR SHARMA – 220905134 – 23 – CSE D
SANJANA.P.S - 220905175 – 27 – CSE D

Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal.

April 2025

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Manipal
09/04/2025

CERTIFICATE

This is to certify that the project titled **Social Media Platform** is a record of the bonafide work done by **AYANIKA PAUL – 220905128, ISHIKA JAISWAL – 220905140, TUSHAR SHARMA – 220905134, SANJANA.P.S - 220905175** submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech.) in COMPUTER SCIENCE & ENGINEERING of Manipal Institute of Technology, Manipal, Karnataka, (A Constituent Institute of Manipal Academy of Higher Education), during the academic year 2022-2023.

Name and Signature of Examiners:

1. **Dr. Roopalakshmi R, Associate Professor, CSE Dept.**

2. **Dr. Rajashree K, Assistant Professor, CSE Dept.**

TABLE OF CONTENTS

ABSTRACT

Social media platforms have transformed how people interact, communicate, and consume content. In today’s digital age, they serve not only as tools for personal connection but also as spaces for community building, content sharing, and even business promotion. With millions of users engaging daily across various platforms, the importance of social media in shaping online behavior and digital culture is undeniable.

This project aims to develop a lightweight yet functional social media web application that encapsulates the essential features of popular platforms while maintaining simplicity and customizability. Built using Django, a robust and scalable Python web framework, the application demonstrates how rapid web development can be achieved without compromising on functionality or security. Core features include user registration and authentication, profile customization, friend (or follow) mechanisms, and a personalized feed populated with posts from connected users. Additionally, the application supports real-time content interactions such as liking posts and viewing dynamic updates, simulating the interactivity users expect from modern social networks.

The project's goal is not just to build a usable application, but also to lay the foundation for a scalable and extendable system that can be enhanced with features like messaging, media sharing, and advanced analytics in future iterations.

CHAPTER 1: INTRODUCTION

The evolution of digital communication has significantly transformed the way individuals interact and share information. Social media platforms have become an integral part of modern society, enabling users to express their thoughts, connect with others, and build virtual communities. With the increasing demand for personalized and privacy-conscious platforms, the need for customizable, open-source, and easy-to-deploy social media solutions is more critical than ever. This mini project aims to bridge this gap by presenting a basic yet powerful social media web application built using Django, a robust Python-based web framework.

This project focuses on building the fundamental features of a social media application while maintaining scalability and extensibility. At its core, the platform allows users to register and authenticate securely using Django's built-in user management system. Upon successful registration and login, users can customize their profile by uploading a profile picture and adding personal information such as a bio and location. The user profile acts as a digital identity, visible to other users on the platform.

A key feature of the application is the ability to create and manage posts. Users can upload images with captions, which are displayed in a chronological feed. This feed dynamically updates to show posts created by the logged-in user as well as those posted by users they follow. Posts can be liked or unliked, and users can engage with content from their network, emulating the interactive nature of contemporary social media apps.

To foster user engagement and network-building, the application implements a follow/unfollow mechanism. Each user can follow others, which tailors their home feed to display content exclusively from those they are interested in. Conversely, they can unfollow users to curate a more personalized experience. This feature encourages organic growth of connections and aligns with user-centric content delivery.

Furthermore, a search functionality has been integrated to allow users to discover other users or posts based on keywords. This enhances navigation and promotes content exploration within the platform. The application also includes an 'Explore' page where all posts are visible regardless of connection status, offering a broader view of community activity.

The project was developed with responsiveness and user experience in mind. The frontend design, while simple, ensures that the interface is clean, intuitive, and compatible across different devices. On the backend, Django’s model-view-template (MVT) architecture ensures clean separation of concerns and allows for rapid development and maintenance.

This report outlines the planning, design, and implementation of the platform, covering the technical stack, database schema, core views and routes, and user experience considerations. The application provides a foundational structure that can be expanded with features such as messaging, notifications, hashtags, or even machine learning-driven recommendations.

In summary, this project successfully demonstrates the feasibility of developing a lightweight, functional, and extendable social media platform using Django. It serves as an educational tool as well as a starting point for more complex systems, making it a valuable contribution to open-source web development and student-level project work.

CHAPTER 2: PROBLEM STATEMENT & OBJECTIVES

Problem Statement

The proliferation of social media has fundamentally changed the way people communicate, collaborate, and share information. From personal networking to business marketing, social platforms have become critical infrastructure in modern society. However, the dominant social media platforms—such as Facebook, Twitter (now X), Instagram, and TikTok—are proprietary, feature-heavy, and not designed with educational or experimental goals in mind. They do not provide access to their backend architectures, making them unsuitable as reference models for students or developers seeking to build their own platforms.

Most available social media codebases online are either outdated, overly simplistic, or heavily abstracted, offering limited educational value. Developers who wish to experiment, extend, or customize features must either start from scratch or attempt to understand a highly convoluted codebase, which often proves to be time-consuming and inefficient. For students in particular, there is a gap between theoretical understanding of web frameworks and practical application in real-world projects.

There is, therefore, a clear need for a minimal, clean, and functional social media platform template that can serve both as a learning tool and as a base for more advanced projects. Such a template should demonstrate essential functionality—user authentication, profile management, content creation, inter-user relationships, and responsive design—while remaining comprehensible and extendable.

The purpose of this project is to create a foundational social media application using Django, a high-level web framework based on Python. Django was chosen for its emphasis on clean, pragmatic design and its robust set of tools for handling user authentication, object-relational mapping (ORM), and admin interfaces. By using Django, this project ensures that the codebase is logically structured, secure, and scalable.

This project aims to simulate the functionality of a basic social media platform, offering the ability to register and authenticate users, create and update profiles, share posts, follow and unfollow users, and interact with content. The resulting application not only serves as a working prototype but also as a highly adaptable base that can be customized or extended based on user requirements or educational goals.

Objectives

To address the identified problem, the project was guided by several key objectives. These objectives were carefully designed to balance practical implementation with educational clarity, ensuring the resulting platform is both functional and instructive.

1. Implementation of Secure User Registration and Authentication

The foundation of any user-based web application lies in its authentication mechanism. This project leverages Django's built-in authentication system to implement secure and user-friendly login, registration, and logout functionalities. Features include:

- Form-based user registration with validation checks.
- Encrypted password storage using Django’s hashing algorithms.
- Session management to persist user logins securely.
- Redirects and user feedback upon successful or failed login attempts.

By utilizing Django’s User model and its authentication views, this objective promotes best practices in handling user credentials, reducing the risk of common vulnerabilities such as session hijacking or brute-force attacks.

2. Development of Profile Management Features

Social media platforms require users to have personalized profiles that represent their digital identity. This project introduces a custom Profile model linked to the default Django User model via a one-to-one relationship. Profile features include:

- Display and update of profile pictures (with a default fallback).
- A biography or description field.
- Optional data such as location, interests, or joined date.

File handling is managed using Django’s ImageField and static/media settings, ensuring that image uploads are handled securely and correctly displayed. This module allows each user to curate a personal presence on the platform and lays the groundwork for advanced personalization features in the future.

3. Creation of a Post System for Sharing Content

At the core of social platforms is the ability to share updates, thoughts, and media. This system enables users to:

- Create posts consisting of an image and a caption.
- Associate posts with users and timestamps.
- Display posts in reverse chronological order on user profiles and feeds.
- Delete their own posts when needed.

The post model captures key information and uses Django’s ORM to query and display data efficiently. This objective also demonstrates how data is passed between views, templates, and models in Django.

4. Implementation of a Follow/Unfollow Mechanism

To foster interaction and content personalization, a follow system was implemented. This system enables users to:

- Follow or unfollow other users.
- View posts from followed users on their personal feed.
- View follower and following counts on profile pages.

This introduces many-to-many relationships in the database and requires careful query filtering to display relevant posts. The system is implemented using a dedicated Follow model and custom view logic to manage user relationships. This objective mirrors real-world features seen in Twitter and Instagram, helping users focus their content consumption.

5. Enabling Post Interactions Through Likes

Interactivity is further enhanced by the ability to like and unlike posts. Features include:

- Users can toggle their like status on any post.
- Each post maintains a like count.
- Like status is reflected visually for each user.

The like system demonstrates conditional database operations and highlights the use of many-to-many fields. It also emphasizes the importance of UX feedback in interactive elements.

6. Ensuring a Clean, Responsive, and User-Friendly Interface

While functionality is paramount, usability cannot be overlooked. The platform’s interface is designed to be:

- Clean and minimal, ensuring focus remains on content.
- Responsive to different screen sizes using Bootstrap or custom CSS.
- Intuitive, with easy navigation between key areas like the profile, feed, post creation, and logout.

Forms and buttons are styled for clarity, and meaningful error/success messages are displayed to guide users. This objective ensures that even a technically strong backend does not fall short in terms of user engagement.

7. Creating a Base for Future Expansion

Although the current scope is limited to essential features, the codebase has been structured with extensibility in mind. Potential future features include:

- Direct messaging between users.
- Comments on posts.
- Notifications and activity tracking.
- Hashtag or tagging systems.

- Advanced privacy settings.

By focusing on modularity and adherence to Django’s best practices, this project establishes a strong foundation for ongoing development.

Conclusion of Objectives

Collectively, these objectives aim to create a comprehensive yet accessible social media platform that satisfies the core needs of a modern social network while serving as a valuable educational resource. The project offers practical insights into web development with Django, including:

- Backend logic and data modeling
- Frontend design integration
- User experience considerations
- Application security
- MVC (Model-View-Controller) design principles

The end result is a minimal but meaningful application that balances performance with clarity and provides a starting point for both learners and developers to explore, build, and innovate.

CHAPTER 3: METHODOLOGY

The development of the social media platform was guided by a systematic and modular approach rooted in the **Model-View-Controller (MVC)** architecture that Django follows, often referred to more specifically as **Model-Template-View (MTV)** in Django terminology. This architecture separates concerns into three distinct layers—data modeling, business logic, and user interface rendering—making the application scalable, maintainable, and easy to debug.

1. Model-Template-View Architecture in Django

- **Models**: Models define the structure of the database. Each model corresponds to a table in the backend database and defines the schema for a specific data object. In this project, several models were created:
  - User: Inherited from Django's default user model, used for authentication and user identity.
  - Profile: Extended user profile containing attributes like profile picture and bio.
  - Post: Represents individual posts containing images and captions.
  - Followers: Represents the follow relationships between users.
  - LikePost: Records which users liked which posts, supporting like/unlike toggling.
- **Views**: Views act as the bridge between models and templates. They handle business logic, data processing, and control the flow of information between the database and the user interface. Key views in the project include:
  - signup(): Handles new user registration and saves valid user data to the database.
  - loginn(): Authenticates users using Django's login mechanisms.
  - logoutt(): Safely logs out users and clears sessions.
  - upload(): Allows authenticated users to create and publish new posts.
  - delete(): Enables users to delete their own posts from the database.
  - likes(): Implements the logic for liking or unliking a post and updating the like count accordingly.
  - follow(): Manages follow and unfollow actions between users, updating the Followers model.
  - profile(): Displays the user's personal profile or another user's profile based on the request.
  - explore(): Lists users not followed by the current user, encouraging network growth.
- **Templates**: Templates are HTML files that dynamically render data from views. These files use Django's templating language to loop through data structures, conditionally display content, and integrate user interactions with form inputs. The templates also include frontend styling using Bootstrap and custom CSS to ensure a clean and responsive user experience.

2. Key Functional Components

- **Authentication System**: Secure user authentication was implemented using Django's built-in forms and authentication framework. CSRF protection, password hashing, session control, and error handling were integrated to ensure data privacy and usability.
- **Posting System**: The upload() function enables users to submit posts with an image and caption. These posts are linked to the user's profile and stored in the database.
  The delete() view ensures only the post creator can remove their content, reinforcing access control.
- **Like Mechanism**: The likes() function uses conditional checks to determine whether a user has already liked a post. If so, it removes the like; if not, it adds it. This ensures that each user can like/unlike posts only once, reflecting real-time interactions on the platform.
- **Follow/Unfollow System**: This functionality is managed through the Followers model. The follow() view toggles the relationship based on user actions. This directly affects the content seen in the user's feed by filtering posts only from followed users.
- **Search Functionality**: A basic search mechanism was added to allow users to discover other users by username. The search bar processes queries and filters users dynamically, enhancing engagement and exploration within the app.
- **Profile and Explore Views**: The profile() view displays posts made by a specific user, their profile details, follower/following stats, and provides follow/unfollow options.
  The explore() view encourages content discovery by listing users not yet followed by the logged-in user.

3. Development Workflow

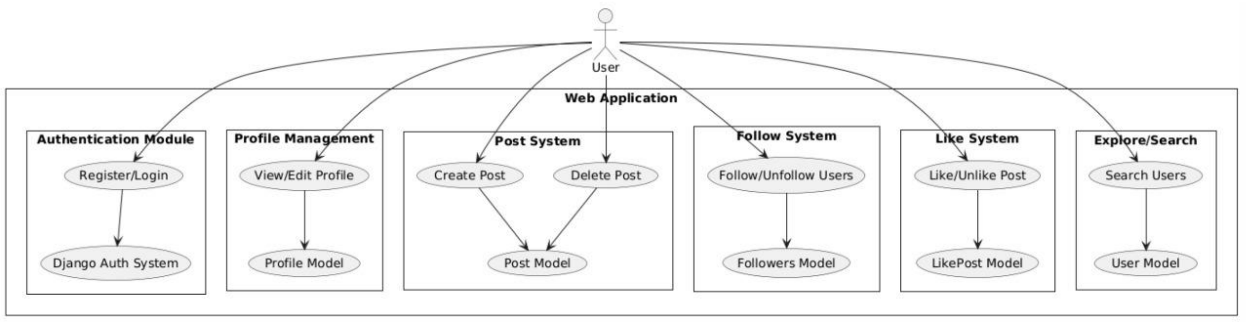
The system was developed incrementally, following an iterative and test-driven development methodology. Version control was handled using **Git** and the code was hosted on **GitHub**, ensuring:

- Safe version tracking.
- Collaboration and branching (if expanded by teams in future).
- Backup and rollback capabilities in case of errors.

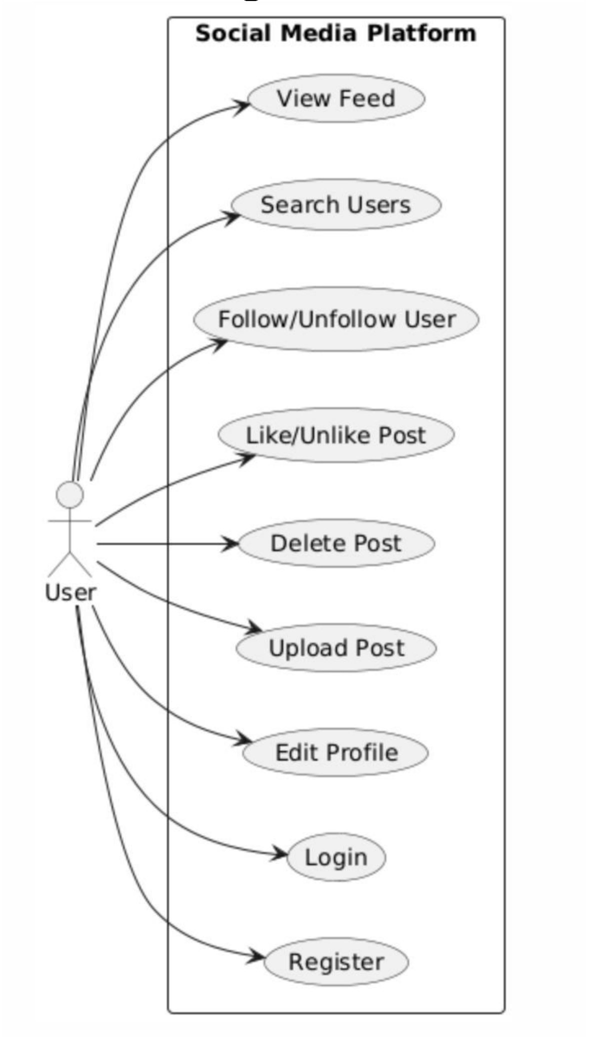
Development was done in a local environment using Django's built-in development server. Frequent testing was performed after the addition of each new feature to ensure stability and correctness. Static files and media uploads were configured using Django's media routing system, and settings were modularized for easier deployment.

This methodical, component-wise approach allowed for progressive building and debugging, ensuring that each module was working independently before being integrated into the full application.

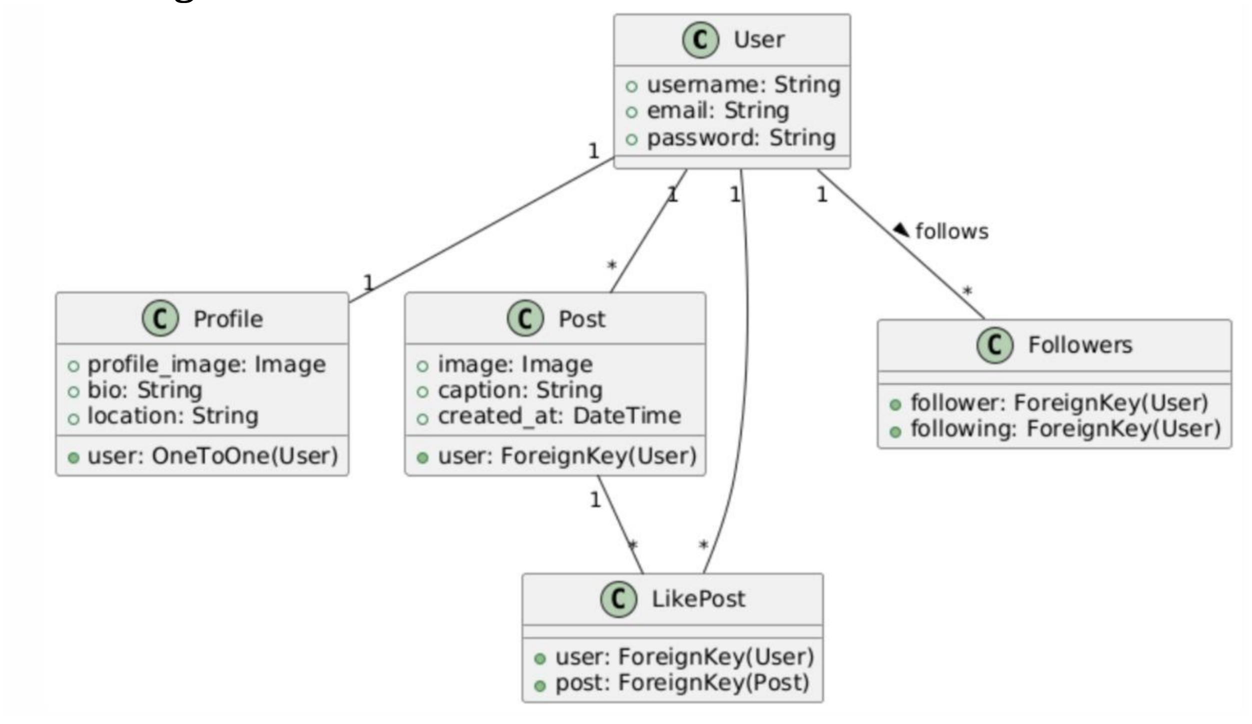
DFD(level 1) :



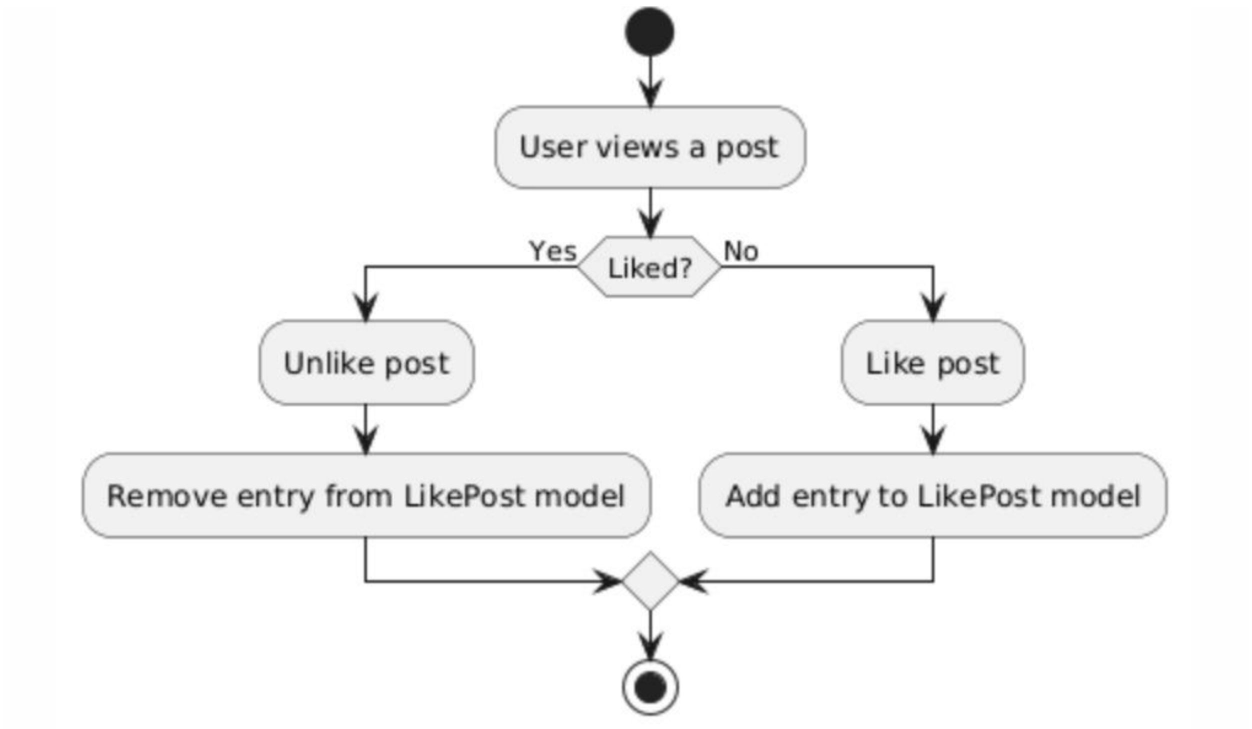
Use Case Diagram :



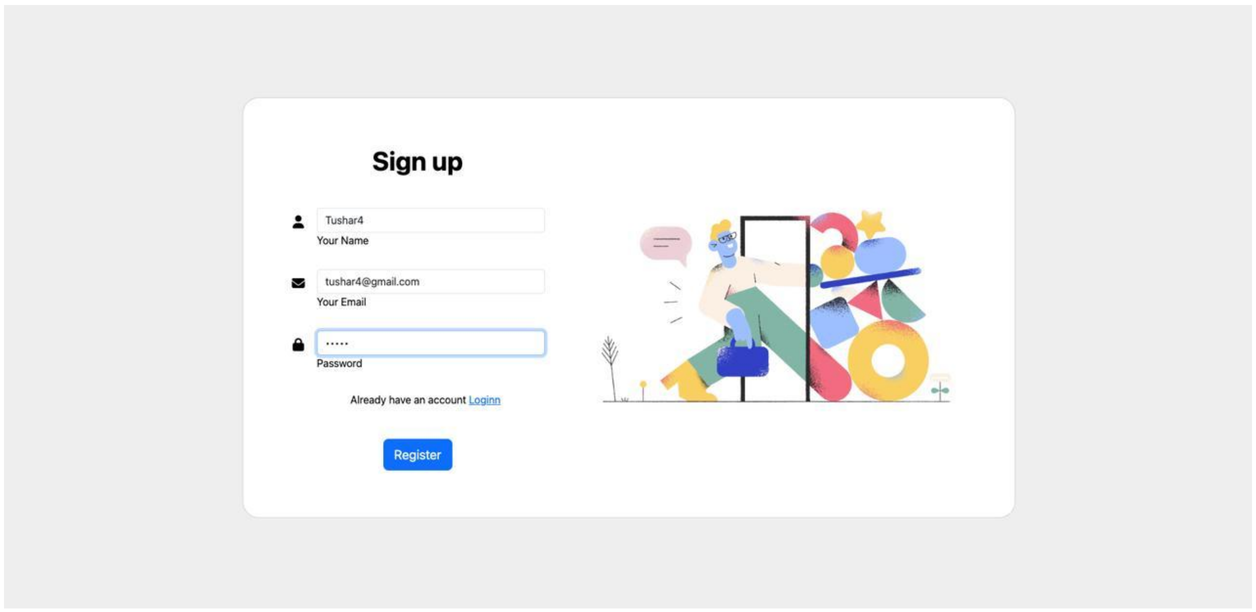
Class Diagram :

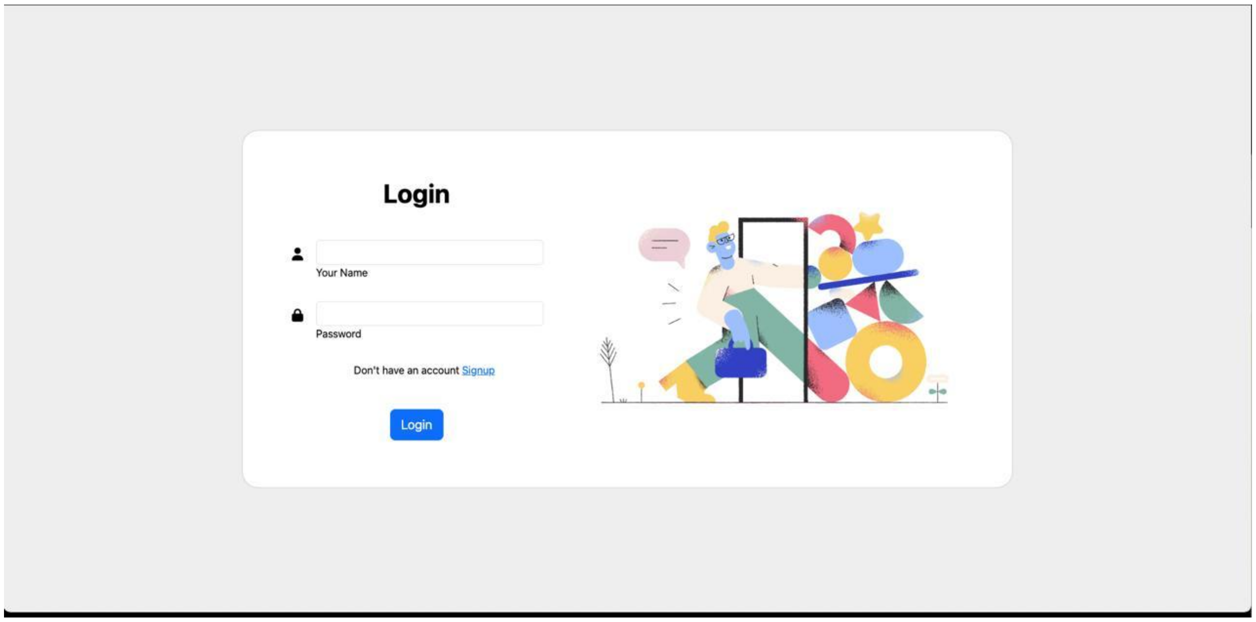


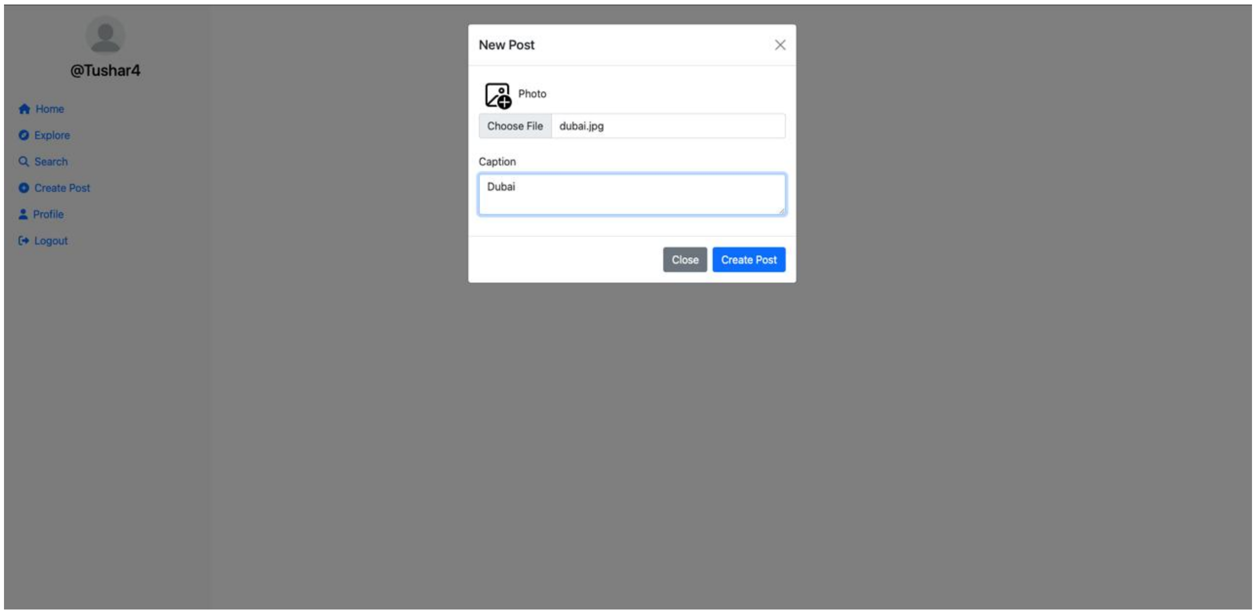
Activity Diagram :



CHAPTER 4: RESULTS & SNAPSHOTS



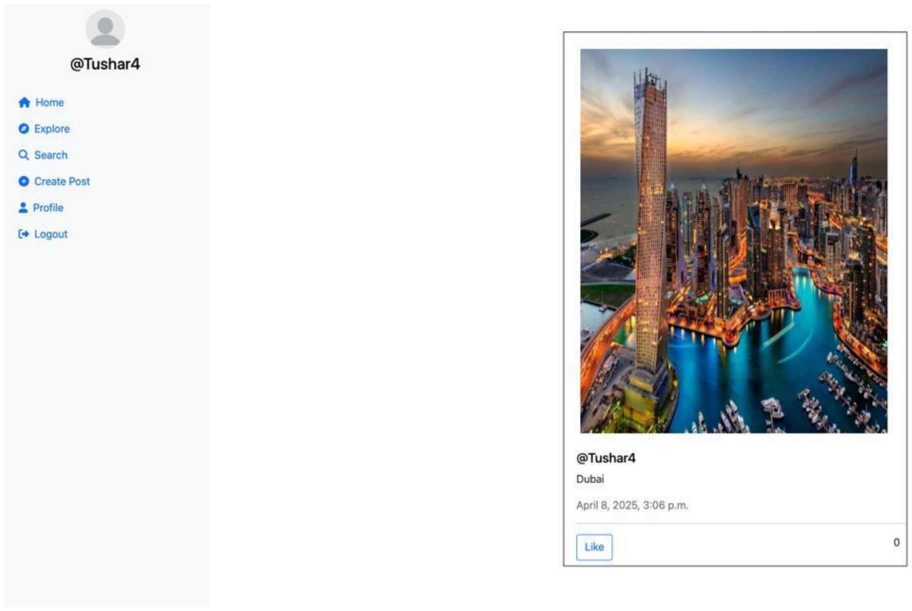




dubai.jpg

Caption

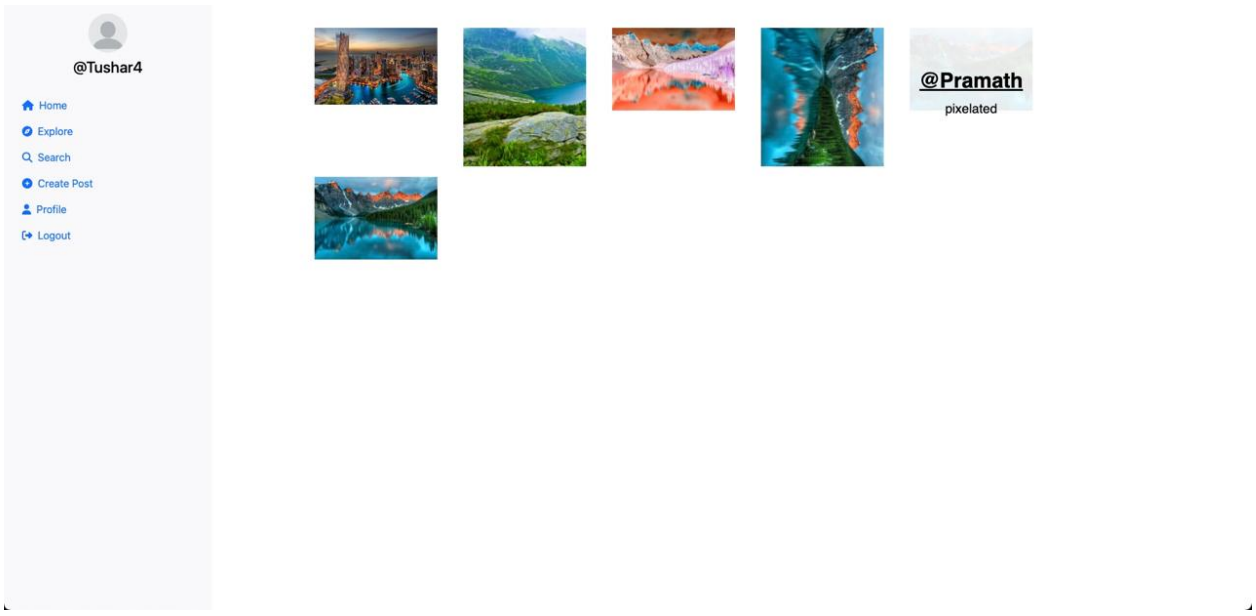
Dubai

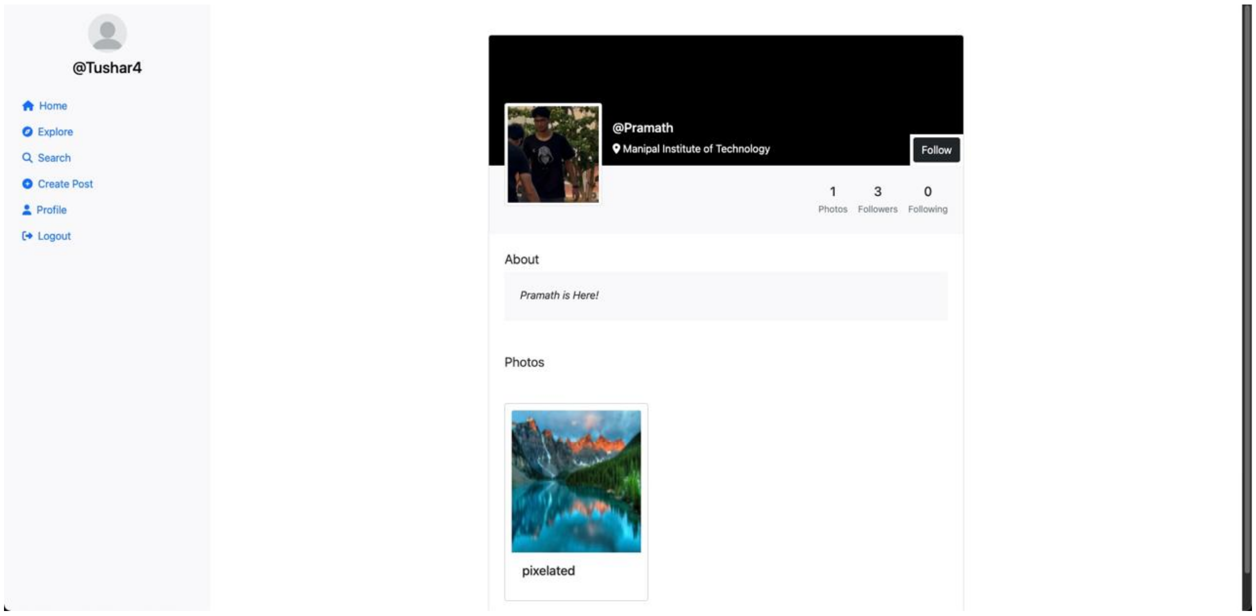


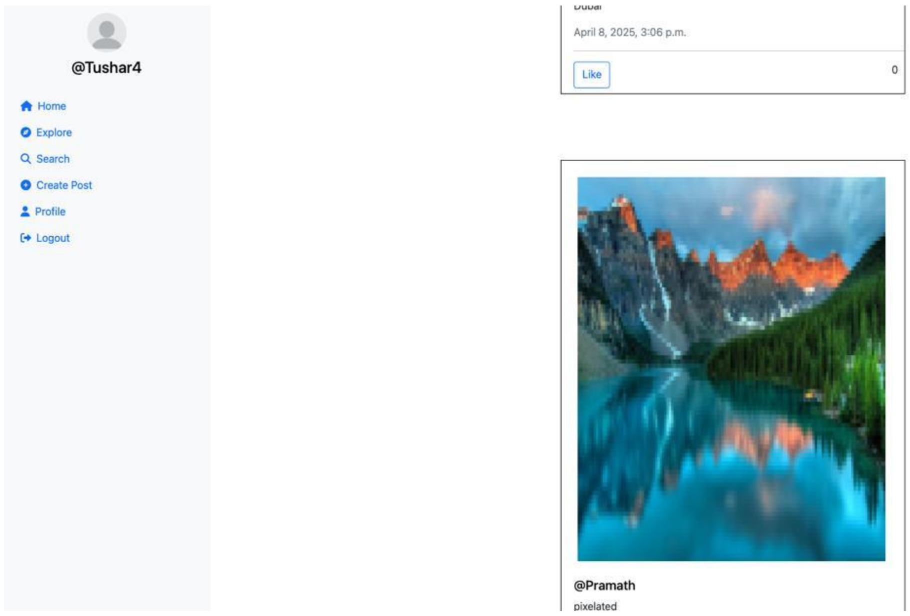
@Tushar4

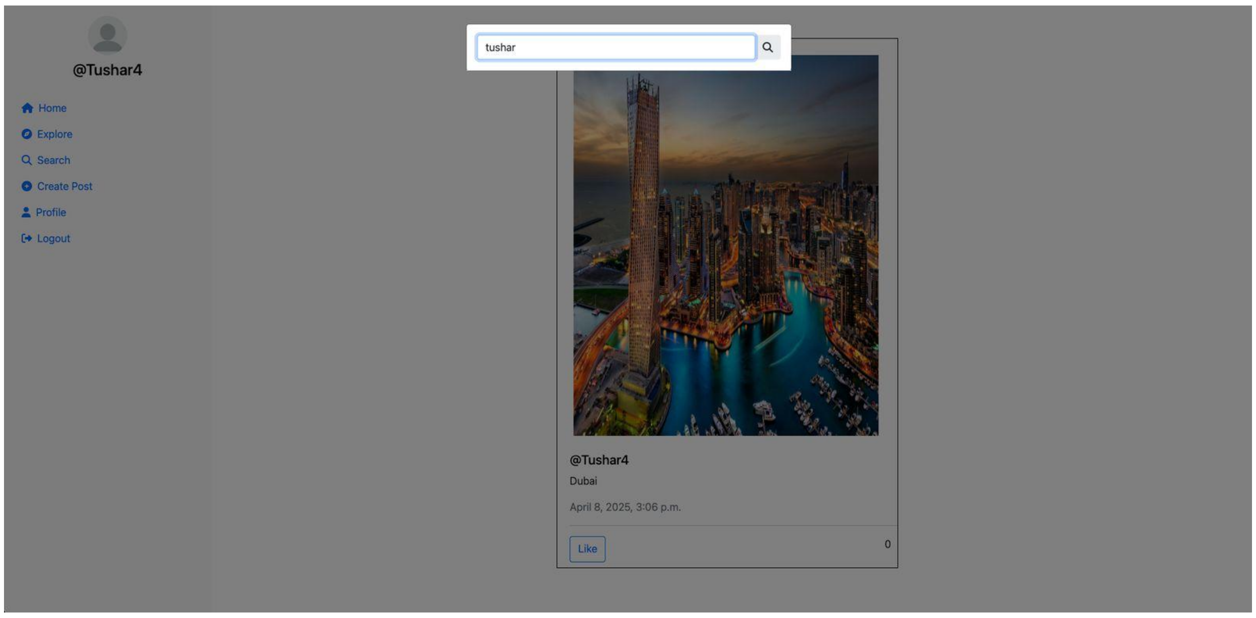
Dubai

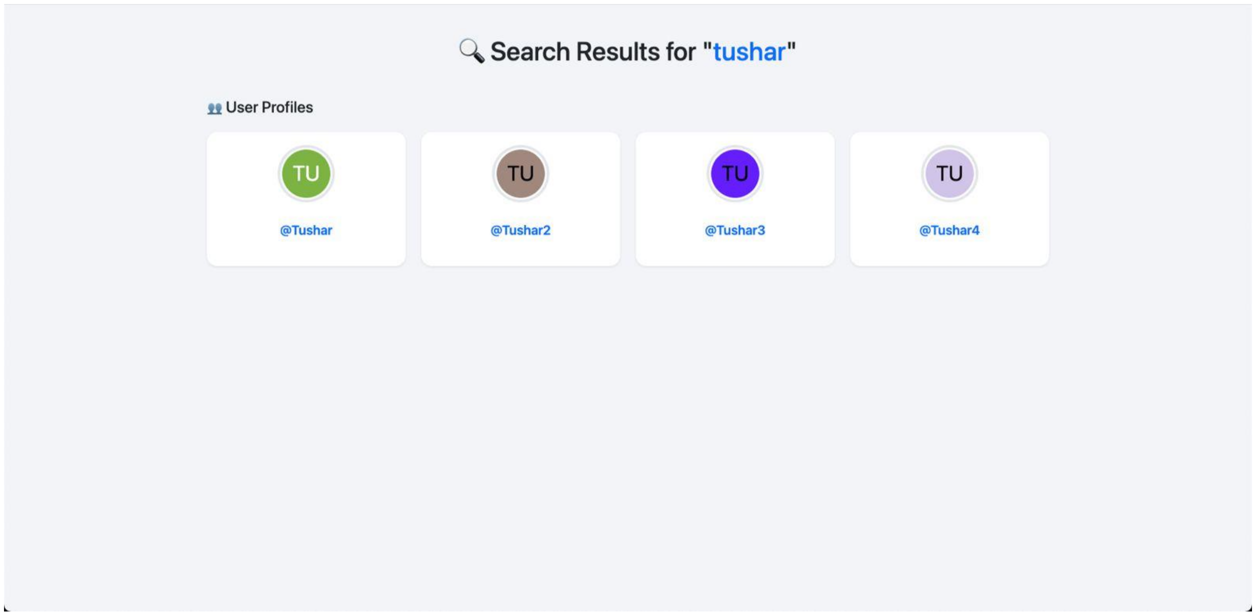
April 8, 2025, 3:06 p.m.


@Tushar4
Home
Explore
Search
Create Post
Profile
Logout
@Pramath
pixelated


@Tushar4
Home
Explore
Search
Create Post
Profile
Logout
@Pramath
Manipal Institute of Technology
Follow
1
Photos
3
Followers
0
Following
About
Pramath is Here!
Photos
pixelated


@Tushar4
Home
Explore
Search
Create Post
Profile
Logout
Dubai
April 8, 2025, 3:06 p.m.
Like
0
@Pramath
pixelated


@Tushar4
Home
Explore
Search
Create Post
Profile
Logout
tushar
@Tushar4
Dubai
April 8, 2025, 3:06 p.m.
Like
0


🔍 Search Results for "tushar"

👥 User Profiles

CHAPTER 5: CONCLUSION

The development of this social media platform using Django demonstrates the capability of open-source web frameworks in building robust, secure, and scalable web applications. Throughout the project, the primary goal was to design and implement a fully functional prototype that mimics the core interactions of modern social platforms while remaining simple enough for educational use and future expansion.

The application successfully integrates all essential features of a social media system, including user registration, login/logout functionality, profile management, post creation with images and captions, the ability to like and unlike posts, and a follow/unfollow system. These features work together to provide a basic but engaging user experience, laying the foundation for a more feature-rich application.

One of the major strengths of this project lies in its **modular and maintainable architecture**. By adhering to Django's Model-View-Template (MVT) pattern, the application maintains a clean separation of concerns. Each feature was developed as an independent module, which greatly simplified testing and debugging during development. This modularity ensures that developers can easily add new components, such as messaging, real-time notifications, media tagging, or comments, without breaking the existing functionality.

In addition, Django's built-in support for security—such as CSRF protection, password hashing, session management, and form validation—was leveraged to protect the application against common vulnerabilities. While the platform was primarily built for educational purposes, these features highlight the framework's readiness for production-grade deployment if further optimized.

Moreover, this project also emphasized **user-centered design**. The interface, developed using Django templates and styled with CSS and Bootstrap, is responsive and intuitive. It adapts to various screen sizes, ensuring accessibility across devices. This is crucial for any modern web application, especially in the social media domain where mobile usage is dominant.

The use of **Git and GitHub** for version control played a significant role in managing code changes, maintaining project history, and enabling collaborative development in the future. Every update, bug fix, or feature enhancement was tracked through commits, ensuring a structured and transparent workflow.

Functionally, the **posting system** forms the core interaction loop. Users can create, view, and interact with posts in a way that reflects common usage patterns on mainstream social platforms. The inclusion of like and follow systems introduces a level of social interaction that makes the platform dynamic and personalized—each user's feed is tailored based on their interests and connections.

From a technical learning perspective, this project provided hands-on experience in backend development, frontend integration, database design, form handling, user authentication, and deployment. It also reinforced best practices in software development such as modularity, code reuse, DRY principles, and consistent error handling.

In conclusion, the project meets its initial objective of delivering a lightweight, extensible, and interactive social media platform. It serves as a strong foundation for further academic exploration or real-world enhancement. With additional work, it can evolve into a complete social network with features like comments, direct messaging, media categorization, content moderation, and analytics. The scalability and clarity of the existing codebase make it well-suited for such growth, proving that powerful web applications can be built using accessible tools and open-source frameworks like Django.

CHAPTER 6: LIMITATIONS & FUTURE WORK

While the developed social media platform successfully implements many core functionalities, it is important to acknowledge the limitations present in the current version and explore avenues for future development. These limitations are natural given the project's scope, time constraints, and educational nature, but they also provide opportunities for meaningful enhancements.

Limitations

One of the primary limitations is the absence of **real-time messaging or notifications**. Modern social media platforms thrive on real-time engagement, whether through instant messages, activity alerts, or content updates. In this project, users are not immediately notified when someone likes their post, starts following them, or interacts with their content. This reduces the sense of immediacy and dynamic interaction, which are key to user retention and engagement.

Another significant limitation is the lack of **email verification during signup**. Currently, any user can register with any email address without validation. This opens the door to potential misuse or spam accounts. Email verification not only improves security but also adds a layer of authenticity to user profiles.

From a design and usability standpoint, the application currently employs **basic frontend styling using Bootstrap**, without integrating advanced UI frameworks or libraries. While this approach keeps the interface simple and functional, it lacks the visual polish, responsiveness, and interactivity that modern users expect. There are no animations, transitions, or client-side validation features that would enhance the user experience.

Moreover, the platform supports only **single-image posts** and does not handle video or other media formats. In a world where visual content dominates social feeds, this limitation significantly restricts user expression. Features like media carousels, video playback, and file compression are not yet part of the system.

Lastly, **security measures remain at a basic level**. The application does not include advanced features like two-factor authentication (2FA), account recovery options, or rate-limiting for login attempts. These are essential in production-level applications where user privacy and data integrity are paramount.

Future Work

To address these limitations, several enhancements have been identified as future work:

- **Direct Messaging**: Implementing a private messaging system would allow users to communicate in real time, greatly increasing platform interactivity and engagement. This could be built using Django models or integrated with WebSocket-based solutions for instant updates.

- **Real-time Features**: By incorporating Django Channels or other WebSocket technologies, the platform can support real-time notifications, live comment updates, and instant interactions—bringing it closer to modern user expectations.
- **Frontend Upgrade**: Integrating a modern JavaScript frontend framework such as React, Vue.js, or Angular would allow the application to function as a Single Page Application (SPA). This shift would enhance performance, reduce page reloads, and provide a smoother user experience with advanced interface dynamics.
- **Enhanced Media Handling**: Adding support for multiple images per post, video uploads, and GIFs would significantly improve the expressiveness of the platform. This would require handling file uploads securely, optimizing media formats, and updating the UI to support rich media content.
- **Email Verification & 2FA**: Incorporating email-based account verification during registration and optional two-factor authentication would improve account security. This is particularly relevant if the platform is scaled to handle sensitive user data or becomes open to public registration.
- **User Feedback & Analytics**: Implementing tools to collect user feedback and monitor usage analytics could guide future design decisions, helping prioritize features based on real-world usage and needs.

By addressing these limitations and implementing the proposed improvements, the platform can evolve from a simple academic prototype into a competitive and secure social media solution. These enhancements would not only improve usability and security but also prepare the application for real-world deployment and scaling.

CHAPTER 7: REFERENCES

1. **Django Documentation: https://docs.djangoproject.com/**
2. **Stack Overflow and Django forums**
3. **MDN Web Docs – HTML, CSS, JS**
4. **W3Schools – Python, Django basics**