

SOCIAL NETWORK WEB APPLICATION

Submitted in partial fulfilment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

SOUMA KANTI GHOSH (Reg. No. 37110730)

ABHISHEK KUMAR (Reg. No. 37110007)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600 119

MARCH – 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)



Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119
www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **SOUMA KANTI GHOSH (Reg. No. 37110730)** and **ABHISHEK KUMAR (Reg. No. 37110007)** who carried out the project entitled "**SOCIAL NETWORK WEB APPLICATION**" under my supervision from **August 2019** to **March 2021**.

Internal Guide

Dr. S. Revathy, M.E., Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **SOUMA KANTI GHOSH** hereby declare that the Project Report entitled “**SOCIAL NETWORK WEB APPLICATION**” is done by me under the guidance of **Dr. S. Revathy, M.E., Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean, School of Computing, Dr. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. S. Revathy, M.E., Ph.D.,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

In recent years, social media have attracted considerable attention. Social media giants like Facebook, Instagram etc., have over billions of users using their applications, but most of these applications in the world wide web uses user's private data to monitor, analyse and sell for their own profit. Moreover, some of the apps have complex privacy settings, not even needed by the user but are enabled by default. This project is based upon a full-fledged Social Network Web Application. The main objective of the application is to solve the issue of too many privacy settings, monitored messaging service and complex user interface (UI) in social network web apps nowadays. Users can use this application to create and customize their own profile, upload and share pictures or status with their friends, which can be liked and commented upon, notification system, with the addition of an easy-to-use UI, lesser privacy settings and unmonitored messaging service to ease the user experience.

TABLE OF CONTENTS

ABSTRACT	v
LIST OF FIGURES	viii

CHAPTER No.	TITLE	PAGE No.
1.	INTRODUCTION	1
	1.1 SOCIAL NETWORK APPLICATIONS	2
	1.2 RESEARCH AND SIGNIFICANCE	3
2.	AIM AND SCOPE	5
	2.1 AIM OF PROJECT	5
	2.1.1 OBJECTIVES	5
	2.2 SCOPES	5
	2.2.1 ADVANTAGES	5
3.	SYSTEM DESIGN & METHODOLOGY	6
	3.1 EXISTING SYSTEM	6
	3.2 SYSTEM CAPABILITIES	8
	3.3 SYSTEM FUNCTIONALITIES	11
	3.4 REQUIREMENT SPECIFICATION	12
	3.5 MODULES	13
4.	SOFTWARE DEVELOPMENT METHODOLOGY	16
	4.1 DESCRIPTION OF DIAGRAM	16
	4.2 ARCHITECTURAL DESIGN	18
	4.3 IDENTIFYING THE USERS	18
	4.4 USE CASES	19
	4.5 EMERGING TECHNOLOGY	19
6.	RESULTS AND DISCUSSION	20

7.	CONCLUSION AND FUTURE WORK	31
	REFERENCES	31
	APPENDIX	33
	A. PAPER ACCEPTANCE MAIL	46
	B. PLAGIARISM REPORT	47
	C. JOURNAL PAPER	36
	D. SOURCE CODE	46

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
4.1	BLOCK DIAGRAM	24
4.2	ACTIVITY DIAGRAM	28
5.1	LOGIN PAGE	34
5.2	REGISTRAION PAGE	35
5.3	HOME PAGE	36
5.4	PROFILE FORM	37
5.4.1	ACCOUNT SETTINGS	37
5.4.2	ACCOUNT PRIVACY SETTINGS	37
5.5	EDIT PROFILE PAGE	38
5.6.1	POST PAGE WITH POST IMAGE	39
5.6.2	POST PAGE WITH STATUS	39
5.7	NOTIFICATIONS TAB	40
5.8	MESSAGING TAB	41
5.9	FRIENDS TAB	42
5.10	POST TAB	42

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

A social network is a network of individuals who meet online to communicate by posting information and images, leaving comments, or sending messages. Participants can expand their personal and business contacts by connecting with others on social networking websites and in applications. What exactly makes people want to connect with each other? People want to share experiences, emotions, and news or just have a conversation with a friend. This desire to share and communicate is driven by interdependence.

In recent years, social media have attracted considerable attention. Social media giants like Facebook, Instagram etc., have over billions of users using their applications, but most of these applications in the world wide web uses user's privacy to monitor, analyse and sell their private data. Moreover, some of the apps have complex privacy settings, not even needed by the user but are enabled by default. Too much unnecessary contents in the page makes the UI of these applications unclean and complex to use.

This web application is based on the idea of a full-fledged social connectivity network similar to that of Facebook, and it aims to solve the issue of too many privacy settings, monitored messaging service and complex user interface (UI) in social media sites nowadays. This application is developed using Django, Python in backend and Bootstrap v4, jQuery in frontend, and is delivered as a Web Application. The Home page of this application is connected to the server through WebSocket connectivity, which utilises real-time server push technology, enabling automatic updates of feeds, notifications, chats without refreshing anything. The messaging service in this application is unmonitored, allowing users to have a private experience while interacting through this application. The UI of this application is created in such a way that users will find easy-to-use. A custom designed Dark Mode UI has been developed with the use of HTML5, CSS3, Bootstrap v4 and jQuery. And on top of it all, it is built

responsive, allowing users to interact with this application through mobile or tablet devices.

1.2 SOCIAL NETWORK APPLICATION

In this era of digitalization, Social Networking is the use of Internet-based social media sites to stay connected with friends, family, colleagues, customers or client. Social Networking can have a social purpose, a business purpose, or both. This full-fledged, scalable social network web application is developed to solve the issue of too many privacy settings, monitored messaging service and complex user interface (UI) in social media sites nowadays. It can be made open-source for the users to interact with this application and have a private experience. There's an Admin panel in this application for maintenance of the system, deleting fraudulent user, handling data related bugs. The maintenance of the database is very essential as it one of the important factors in social network application. The database stores all kinds of user-related data, from activity to session data, posts and user profile data including user info and privacy settings. Proper authentication is set-up in this server by default to keep the database safe and to prevent from any loss or damage of the database. Only the authorized person or the admins can access the database by giving the proper log-in credentials, so that the confidentiality and security is maintained.

There are bunch of other functionalities in this web application, including real-time notification tab, real-time friends list with online notifier, adding post or status functionality, search user functionality and an unmonitored messaging service. Functionalities like notification system, friends list and conversations in messaging tab are not loaded until user clicks on the modal buttons in the home page which is used to open these tabs, All these tabs are loaded for the first time and then these tabs are refreshed from server side using WebSocket push technology, i.e., if there's new content available, it will be updated automatically on the client side through server, if the user is online. The user feed section working mechanism is similar to the tabs above, but during the first page load, it gets loaded too. The images in the feed are loaded dynamically (Lazy loading) while the user scrolls through the feed, which helps in faster loading of the home page. A lower resolution of post images are stored in the Post Model for faster loading of post thumbnails in the user profile page.

In this application we have used PostgreSQL for building the database. The User interface is built by using HTML5, CSS3, Bootstrap v4 with jQuery. The backend and the main framework of the program are done using Django with Django Channels extension, which extends Django's abilities to handle other connections in either a synchronous or asynchronous style. A Python based framework "Celery" is being used to handle any sort of background task handling like saving unseen messages in the database, sending notifications to a user etc., Keeping in mind the security of this application we have assigned the authentication and authorization work to the admin who will be able to make changes in the system, the user needs to first register for log-in and then use it. "Whitenoise", A Python based framework is used to handle all the static files used in this application, including the .css and .js files along with any images. This application's user connectivity is similar to that of Instagram, user A can follow another user B, for instance and see B's posts in his/her feed, the followed user B can also follow back in order to see user A's post. Users can make their profile private, so no user can randomly follow them, as they get the authority to accept or reject the follow request of a user.

1.3 RESEARCH AND SIGNIFICANCE

The primary aim of this proposed system i.e. "Social Network Web Application" to solve the issue of too many privacy settings, monitored messaging service and complex user interface (UI) in social media applications. There are several intents of this proposed system. Some of the important intents are as follows:

- This model provides responsive UI which can be used to interact with this application using smaller screen devices.
- This system has database with several modules interconnected with each other for proper structuration of user data.
- It makes the whole process efficient and fast by using WebSocket connectivity, lazy loading of post images, server-side automatic content updates.
- This process is based on digitalization.
- It provides authentication and keeps the system secured.

The proposed model "Social Network Web Application" is based on the new technologies which makes the whole process efficient. It is going to be very efficient

and easy-to-use for the users interacting with this application. This system will provide a private experience for the users, more and more features can be added to this application as the application is made to be scalable. This system has its own admin panel which can be used to control the data being circulated throughout the application. Thus, this system makes the whole process very easy and increase the flexibility of the system. This application will help to decrease the number of privacy settings to only the ones absolutely required by the user. Social Networking applications are changing day by day, adding different features like private and public 24-hours stories and livestreams. These features can be added in this application in future, with the help of smaller tweaks and added modules, packages.

This model will help in building the strategies to prevent analysis and trading of user's private data. This system has an unmonitored messaging service, which helps users to converse freely, in future a setting to save chats will be added in this application for users to keep their chats stored in the database. Now-a-days people feel much unsecured using conversation mechanisms in social network giants like Instagram and Facebook as they monitor everything, moving back to end-to-end encrypted chatting applications like WhatsApp and Telegram. But these applications lack other features which Instagram or other applications offer, like post feeds, discover feed features, stories etc., This application provides a simpler way to socialize, interact with others without all the hassles other social networking application adds to their experience.

CHAPTER 2

AIM AND SCOPE

2.1 AIM OF PROJECT

The primary aim of this proposed system i.e. “Social Network Web Application” is to solve the issue of too many privacy settings, monitored messaging service and complex user interface (UI) in social networking sites nowadays. The proposed system also provides a convenient method for the users to use this application to create and customize their own profile, upload and share pictures or status with their friends, which can be liked and commented upon, notification system, with the addition of an easy-to-use UI, lesser privacy settings and unmonitored messaging service to ease the user experience.

2.1.1 OBJECTIVES

The proposed model “Social Network Web Application” is based on the new technologies which makes the whole process efficient. It is going to be very efficient and easy-to-use for the users interacting with this web application. This system will provide an easy-to-use, attractive Dark Mode UI, lesser privacy settings and an unmonitored messaging service which will help the users using this application to have a private experience.

2.2 SCOPE

This model will help in building the strategies to prevent analysis and trading of user’s private data. This system has an unmonitored messaging service, which helps users to converse freely with other users and have a private experience, along with a responsive, clean and attractive Dark Mode based UI to keep the user hooked in the application.

2.2.1 ADVANTAGES

- Lesser account and account privacy settings to reduce complexity.
- This model provides responsive UI which can be used to interact with this application using smaller screen devices.
- Secure and private environment with unmonitored messaging service.

- Efficient and fast page interaction by using WebSocket connectivity, lazy loading of post images, server-side automatic content updates.

CHAPTER 3

SYSTEM DESIGN & METHODOLOGY

3.1 EXISTING SYSTEM

Now-a-days people feel much unsecured using conversation mechanisms in social network giants like Instagram and Facebook as they monitor everything, moving back to end-to-end encrypted chatting applications like WhatsApp and Telegram. But these applications lack other features which Instagram or other applications offer, like post feeds, discover feed features, stories etc.,

This social network application provides a web based social network application developed using a Python based backend framework, Django along with Channels, which extends Django's abilities to handle other connections in either a synchronous or asynchronous style. In this paper, A Python based framework "Celery" is being used to handle any sort of background task handling like saving unseen messages in the database, sending notifications to a user etc., Along with that, PostgreSQL is used for data storage, Redis is used for caching and also to be a message broker. In the frontend section, the UI is developed to be as clean as possible, with minimal animations, using Bootstrap as a frontend framework and utilising its overlay feature – "Modals" for different sections in the application, from notifications panel to user messaging section.

The main objective of the paper is to implement a fast & efficient, social network web application with minimal privacy settings, unmonitored messaging service and a clean, lightweight UI. The main page or the home page of the application which opens up right after signing in is connected to the server through WebSocket connection, handled by Channels, which makes asynchronous requests between server and client, messaging service possible. Other essential features of this application consist of uploading status or pictures, liking and commenting on posts, bookmarking posts, private and public account settings, accepting or declining friend

requests, personal profile page with posts and bookmarks gallery, edit profile section, changing password section as well.

This model will help in building the strategies to prevent analysis and trading of user's private data. This system has an unmonitored messaging service, which helps users to converse freely, in future a setting to save chats will be added in this application for users to keep their chats stored in the database. Now-a-days people feel much unsecured using conversation mechanisms in social network giants like Instagram and Facebook as they monitor everything, moving back to end-to-end encrypted chatting applications like WhatsApp and Telegram. But these applications lack other features which Instagram or other applications offer, like post feeds, discover feed features, stories etc., This application provides a simpler way to socialize, interact with others without all the hassles other social networking application adds to their experience.

In this application we have used PostgreSQL for building the database. The User interface is built by using HTML5, CSS3, Bootstrap v4 with jQuery. The backend and the main framework of the program are done using Django with Django Channels extension, which extends Django's abilities to handle other connections in either a synchronous or asynchronous style. A Python based framework "Celery" is being used to handle any sort of background task handling like saving unseen messages in the database, sending notifications to a user etc., Keeping in mind the security of this application we have assigned the authentication and authorization work to the admin who will be able to make changes in the system, the user needs to first register for log-in and then use it. "Whitenoise", A Python based framework is used to handle all the static files used in this application, including the .css and .js files along with any images. This application's user to user connectivity is similar to that of Instagram, user A can follow another user B, for instance and see B's posts in his/her feed, the followed user B can also follow back in order to see user A's post. Users can make their profile private, so no user can randomly follow them, as they get the authority to accept or reject the follow request of a user.

3.2 SYSTEM CAPABILITIES

The proposed system i.e. Social Network Web Application consist of various features which makes this model efficient for the user. The users using this application will need to register to create their user profile in order to interact with other users using this application. This application's user to user connectivity is similar to that of Instagram, user A can follow another user B, for instance and see B's posts in his/her feed, the followed user B can also follow back in order to see user A's post. Users can make their profile private, so no user can randomly follow them, as they get the authority to accept or reject the follow request of a user. This system has an unmonitored messaging service, which helps users to converse freely, in future a setting to save chats will be added in this application for users to keep their chats stored in the database. The main page or the home page of the application which opens up right after signing in and is connected to the server through WebSocket connection, handled by Channels, which makes asynchronous requests between server and client, real-time messaging service possible.

In this application we have used PostgreSQL for building the database. The User interface is built by using HTML5, CSS3, Bootstrap v4 with jQuery. The backend and the main framework of the program are done using Django with Django Channels extension, which extends Django's abilities to handle other connections in either a synchronous or asynchronous style. A Python based framework "Celery" is being used to handle any sort of background task handling like saving unseen messages in the database, sending notifications to a user etc., Keeping in mind the security of this application we have assigned the authentication and authorization work to the admin who will be able to make changes in the system, the user needs to first register for log-in and then use it. "Whitenoise", A Python based framework is used to handle all the static files used in this application, including the .css and .js files along with any images.

The users interact with the frontend of the application, any type of request from the client is sent to the server for a response. The server accepts two type of connection requests, HTTP and WebSocket connectivity. The server side is developed using Django with Channels extension to handle WebSocket connections, which sends a response to the client based on the request. For any kind of background task handling, the server side connects with Celery workers, for data storage the server connects with PostgreSQL Database used for this

application and for caching purposes, in-memory cache Redis is used. Redis is also used by Channels and celery as a message broker. The features of the proposed system make it a different one from the traditional method. The proposed model has different section. The sections are as follows:

Admin section

This section is known as admin page which controls the portal, maintains the database and makes necessary changes whenever required. It gives access, authentication and authorization to the users. An Admin user or superuser authorization is required to access this section of the application, there are a whole lot of functionalities in the admin panel provided by Django server, like CRUD (Create, read, update and delete) functionalities in every database model registered to the admin panel. Maintenance of the database is one of the most important aspect of the admin panel.

Login/Signup Section

This section handles the registration and log-in of any user into the application. When the login button is clicked, an AJAX POST request is sent to the server with the login credentials, the server checks the credentials and on success, the user gets logged-in and is redirected to the home page, else incorrect username or password message is displayed to the user. In the signup page of the application. Each of the fields are server-side validated in real-time using AJAX. Celery is used to validate these fields, for example, the username is validated as if it is unique or are there any other user with the same username. Upon successful sign-up, the page gets redirected to home page, and upon unsuccessful sign-up, if there's any kind of validation error in any of the sign-up fields, it gets shown under the field and the field's underline border changes its colour to red.

Home Page Section

This page is connected to the server through WebSocket connectivity. All the user's and user's friend's recent posts are shown in the left section of the Home page, and all the Modal buttons on the right side of the page, the first modal is the notification tab, second one is the friends tab where all the user's following and followers are

shown, the third one is the post tab where the user can post pictures or status, the fourth one is the search tab used to search for an user by their username, and the final one is the messaging tab. The top rightmost section has the link to user's profile section and the logout button. In the notifications tab, all the types of notifications are shown to the user. As this page is connected to the server through WebSocket connectivity, this tab gets updated on real-time. The bottom button in the tab is used to clear all the notifications. In the messaging section, all the conversations are shown, the user can only send or receive messages from users he/she is friends with. The user can chat with only 1 user at a time. Unseen messages are saved in the database until it gets read, then it gets deleted from the server. In the friends list section, the user followed or being followed by the logged-in user gets displayed, and the users in this list being online is highlighted with a green dot on the right of their username. In the upload section, the user gets the opportunity to upload status or pictures with caption and location and post it on their feed. On click to post button, a POST request gets sent to the server, server-side validation including image format checks etc., are checked and on success the page gets refreshed with new post added on user's feed.

Profile Section

This section handles user data, from user bio to user posts and bookmarks, user account settings to user friends list. In the user-profile page, the rightmost 2 buttons which are the account settings modal and the edit profile modal, the first image is user's profile picture, then comes the user's username, then user's bio, then comes the count of user's post, followers and following. The bottom section of the page which consists of user's post gallery and the posts that user has bookmarked.

The account privacy settings section is made as minimal as possible. This includes disabling or enabling notifications of post likes, comments and comment likes, follower notifications, account privacy including enabling or disabling private account option. The edit profile section includes changing profile photo, username, full name, email, birthdate, gender, bio as well as changing password. On clicking the update button, an AJAX POST request is sent to the server for validation and update, on success a notification is sent in real-time that the profile is successfully updated.

3.3 SYSTEM FUNCTIONALITIES

The social network web application model has different types of functions. A straightforward working methodology is incorporated so the client can comprehend the distinctive capacities obviously and rapidly. At first as an initial step the executable type of the application is to be made and stacked in the regular server machine which is available to the whole client and the server is to be associated with a system. In this application we have used PostgreSQL for building the database. The User interface is built by using HTML5, CSS3, Bootstrap v4 with jQuery. The backend and the main framework of the program are done using Django with Django Channels extension, which extends Django's abilities to handle other connections in either a synchronous or asynchronous style. A Python based framework "Celery" is being used to handle any sort of background task handling like saving unseen messages in the database, sending notifications to a user etc., Keeping in mind the security of this application we have assigned the authentication and authorization work to the admin who will be able to make changes in the system, the user needs to first register for log-in and then use it. "Whitenoise", A Python based framework is used to handle all the static files used in this application, including the .css and .js files along with any images.

There are bunch of other functionalities in this web application, including real-time notification tab, real-time friends list with online notifier, adding post or status functionality, search user functionality and an unmonitored messaging service. Functionalities like notification system, friends list and conversations in messaging tab are not loaded until user clicks on the modal buttons in the home page which is used to open these tabs, All these tabs are loaded for the first time and then these tabs are refreshed from server side using WebSocket push technology, i.e., if there's new content available, it will be updated automatically on the client side through server, if the user is online. The user feed section working mechanism is similar to the tabs above, but during the first page load, it gets loaded too. The images in the feed are loaded dynamically (Lazy loading) while the user scrolls through the feed, which helps in faster loading of the home page. A lower resolution of post images is stored in the Post Model for faster loading of post thumbnails in the user profile page.

Some of the functions are given below:

- WebSocket connected Home page which includes real-time notification tab, real-time friends list with online notifier, adding post or status functionality, search user functionality and an unmonitored messaging service.
- The user can post images with captions and location or status without an image (Text only).
- User Edit Profile panel including completing edit of profile data, changing password while logged-in.
- Bookmarking, liking and commenting on posts, also deletion of user's own posts.
- An Admin panel to maintain everything in the application.
- Real-time Unmonitored messaging service to allow full privacy while conversing with a friend.

3.4 REQUIREMENT SPECIFICATION

This proposed software runs effectively on a computing system that has the minimum requirements. Undertaking all the equipment necessities are not satisfied but rather exist in their systems administration between the customer's machines already. So, the main need is to introduce appropriate equipment for the product.

The requirements are split into two categories, namely:

Software Requirements

The basic software requirements to run the program are:

- i. Linux
- ii. Python v3 or higher
- iii. Django
- iv. Django Channels
- v. Celery
- vi. PostgreSQL
- vii. Redis
- viii. Whitenoise

Hardware Requirements

The basic hardware required to run the program are:

- i. Hard disk of 20 GB
- ii. System memory (RAM) of 1 GB or higher
- iii. i3 processor-based computer or higher.

3.5 MODULES

The social network web application model consists of mainly three 4 modules. They are Admin, User, Post and Friends module. The primary task of the admin is to maintain the database records and manage everything related to content in the system. The User module is used to save, search, delete user related data in the database and also for to relate to different modules dependent on this one as a foreign-key relationship or many-to-many relationship or one-to-many. The post module is used to save, search, delete posts related data in the database, directly related to the User module by a Foreign key. Datapoints like post date and time, post likes and comments counts are stored and can be queried via this module. The Friends module handles user's friend lists, having many-to-many relations with the User module to maintain user-to-user connections, moreover having bunch of utilities which are used to form or delete connections between users.

There are four important modules. They are mentioned below:

Admin Module

This module keeps the whole system secure and it doesn't allow any unauthorized users to access the system. The primary task of the admin is to maintain the database records and manage everything related to content in the system. Only those user accounts with superuser privileges can access this section of the system. The admin has the capabilities to use all the models to automatically build a site area that can be used to create, read, update and delete (CRUD) records, making it pretty easy to test out models and get a feel for whether the right data is been stored. For instance, in this application the admin can delete a user who is

posting malicious content in the system, change a post's caption if they find it inappropriate etc., The admin panel is a default extension given by Django, the modules which needs to be maintained are to be manually registered in the admin panel by editing the admin.py file.

User Module

This module is used to save, search, access and delete user related data in the system and can be used to query different datapoints related to user like a particular user's post or friend list. This module has its own default authentication mechanism provided by Django, and is customizable to user's need. In this application, the User module stores user's info and settings, profile data and is related to Post module and Friends module. Datapoints like username, email, superuser permission, password hash, session key etc., are stored by this module. The session key stored by this module can be used to access user's session data, the superuser permission is stored by the datapoint "is_admin" which is used to distinguish normal users from admin users in the application. This module is directly converted into User table in the database, with the datapoints acting as columns in the database, saving different data. As this module is directly connected to the database, it becomes easier to query, update and delete records from the database.

Post Module

This module is used to save, search, delete posts related data in the database, directly related to the User module by a Foreign key. This module stores post data including post image location, post caption, post like and comment counts, post date & time etc., This module is connected to 2 sub-modules PostLikes and PostComments which are used to handle post likes and comments respectively. This module is registered to the admin panel for the admins to monitor posts being circulated across the application. This module is directly converted into User table in the database, with the datapoints acting as columns in the database, saving different data. As this module is directly connected to the database, it becomes easier to query, update and delete records from the database.

Friends Module

The Friends module handles user's friend lists, having many-to-many relations with the User module to maintain user-to-user connections, moreover having bunch of utilities which are used to form or delete connections between users. This module has a Foreign key relation with the User module, followers and following list have a many-to-many relationship with the User module, through this module the list of friends and pending friend requests of a particular user is accessible. This module is directly converted into User table in the database, with the datapoints acting as columns in the database, saving different data. As this module is directly connected to the database, it becomes easier to query, update and delete records from the database. This module has bunch of utilities to help the connection process between 2 users, classmethod follow is used to follow an user, classmethod unfollow is used to unfollow an user, classmethod "add_to_pending" is used to add a friend request to pending queue i.e., still to be accepted or rejected by an user who has set his/her account to private mode. Similarly, classmethod "rm_from_pending" is used to remove a friend request from pending queue that means the user has either accepted or rejected the friend request. Finally, a classmethod "get_friends_list" is also there to get the list of followers and following of a user.

CHAPTER 4

SOFTWARE DEVELOPMENT METHODOLOGY

4.1 DESCRIPTION OF DIAGRAM

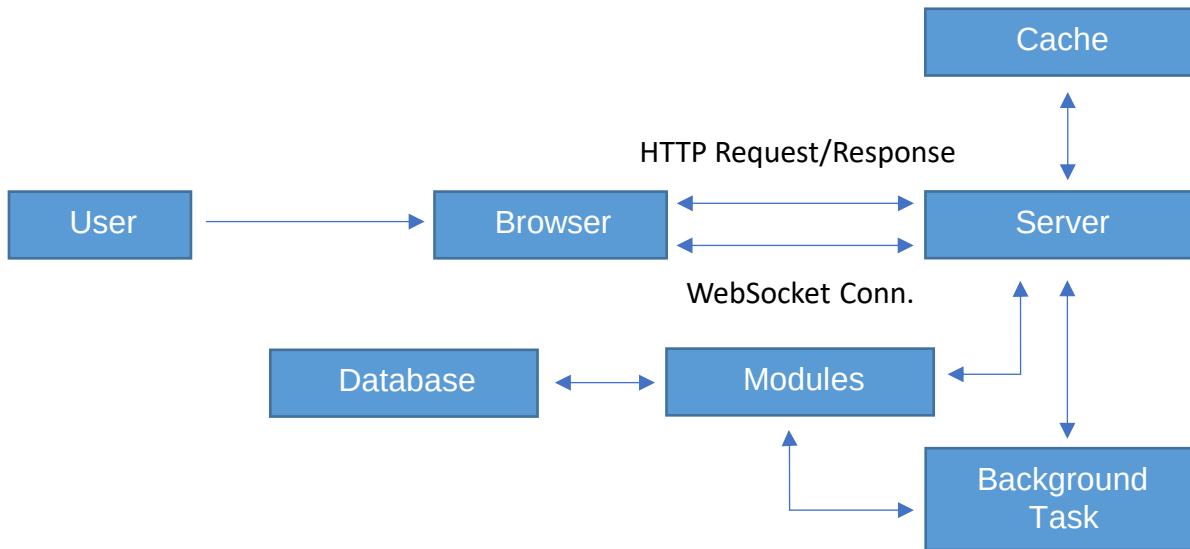


Fig 4.1: BLOCK DIAGRAM

In the Fig 4.1, the flow of data provided to and received from the “Social Network Web Application” is identified. The arrows represent the flow of data received or generated by the client and the server. The closed boxes represent the set of sources and sinks of data. Within the system, we will observe that the user interacts with the application through a browser. The browser sends and receives requests and responses to and from the server, the server can use caching and other connected features depending on the type of request.

The server is developed in such a way that it can handle two types of connection request, the first one being the traditional HTTP request and the second one being a WebSocket connection request. Django channels extension is used to handle the WebSocket connectivity, which is used only for the Home page as it needs to deliver data in real-time (Notifications, Posts, Chats etc.,)

For any kind of database related job, the server uses the ORM (Object Relational Mapper) Modules to query the database easily using the Python language. The advantage of ORM is to interact with the database easily and efficiently thereby eliminating the hassle for developers. The server caches any kind of data to be used repeatedly, for background task handling like sending notifications, saving unseen conversations, the server interacts with Celery, A Python based framework used to handle any sort of background task handling. The Task handler also uses the ORM Modules to interact with the database for any database related jobs. For caching and message broker purposes, Redis is used in this application, For data storage “PostgreSQL” is used, For server side development, “Django” with “Django channels” extension is used, For serving static files like css and js files used in the frontend section, “Whitenoise” is used. The frontend or the client-side development section is built using HTML5, CSS3, jQuery with Bootstrap v4 framework. This side is visible to the users, A JS framework Reconnecting WebSocket is used to connect to the server through WebSocket, one advantage of using this extension is if the connection fails, this extension ensures that it tries again and again to connect with the server after a few seconds of delay between requests. AJAX is also been used in some of the sections like in the login page and the registration page, where WebSocket connection seemed like an overkill.

There are bunch of other functionalities in this web application, including real-time notification tab, real-time friends list with online notifier, adding post or status functionality, search user functionality and an unmonitored messaging service. Functionalities like notification system, friends list and conversations in messaging tab are not loaded until user clicks on the modal buttons in the home page which is used to open these tabs, All these tabs are loaded for the first time and then these tabs are refreshed from server side using WebSocket push technology, i.e., if there's new content available, it will be updated automatically on the client side through server, if the user is online. The user feed section working mechanism is similar to the tabs above, but during the first page load, it gets loaded too. The images in the feed are loaded dynamically (Lazy loading) while the user scrolls through the feed, which helps in faster loading of the home page. A lower resolution of post images is stored in the Post Model for faster loading of post thumbnails in the user profile page.

4.2 ARCHITECTURAL DESIGN

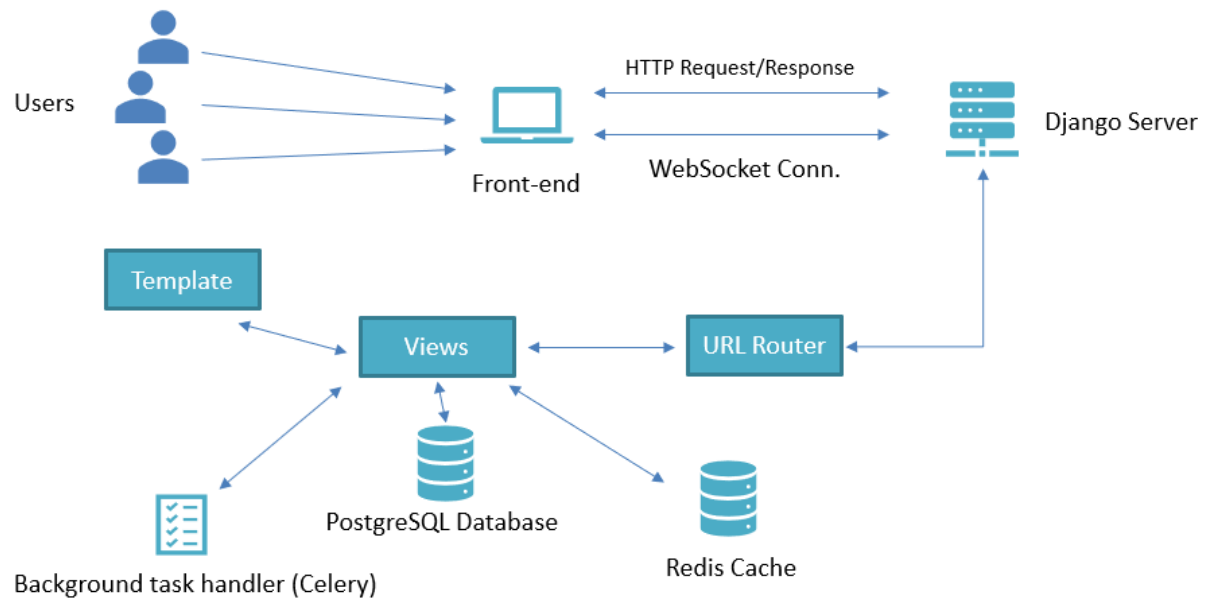


Fig 4.2: Activity Diagram

4.3 IDENTIFYING THE USERS

The users within the system are the admins and the client, the admins keep the whole system secure and it doesn't allow any unauthorized users to access the system. The primary task of the admin is to maintain the database records and manage everything related to content in the system. The client in this application will seek to have a private experience while socializing with other clients in the same application. To interact with other clients, they can follow each other, converse through an unmonitored messaging service, like and comment on each other's posts.

4.4 USE CASES

A social network is a network of individuals who meet online to communicate by posting information and images, leaving comments, or sending messages. Participants can expand their personal and business contacts by connecting with others on social networking websites and in applications. In Social Network Web Application system, there are various types of users. Take each user and identify the roles played, which might lead us to spot the roles played, an identification of use case. The proposed system has different users, different roles and different use cases for instance of identifying the utilization cases, A user might play multiple roles.

The roles are listed below:

1. Educational social networking let users (students) collaborate and include bookmarking and content curation networks. The bookmark post feature in this application can be used for that purpose.
2. Social connections network lets users stay in touch with each other. Friends module is used for this purpose.
3. This application might be useful for users to create a professional network.

As explained above, the roles are use cases. Similarly, any user can use the cases in this application.

4.5 EMERGING TECHNOLOGY

In this world of technology the world is reaching to newer heights through innovation and technology day-by-day. Technology has changed the lives of the people and made it easier. So we must take the advantage of the technology and apply it for the well-being of the society. So we have decided to make use of the cutting edge technologies to build this application which will help the users to use this application and a private experience. In our system In this application we have used PostgreSQL for building the database. The User interface is built by using HTML5, CSS3, Bootstrap v4 with jQuery. The backend and the main framework of the program are done using Django with Django Channels extension, which extends Django's abilities to handle other connections in either a synchronous or

asynchronous style. A Python based framework “Celery” is being used to handle any sort of background task handling like saving unseen messages in the database, sending notifications to a user etc., Keeping in mind the security of this application we have assigned the authentication and authorization work to the admin who will be able to make changes in the system, the user needs to first register for log-in and then use it. “Whitenoise”, A Python based framework is used to handle all the static files used in this application, including the .css and .js files along with any images.

CHAPTER 5

RESULTS AND DISCUSSION

This section shows the outcome of the proposed work of the project “Social Network Web Application”. The client side view of the application along with responses have been captured and the methodology has been properly explained.

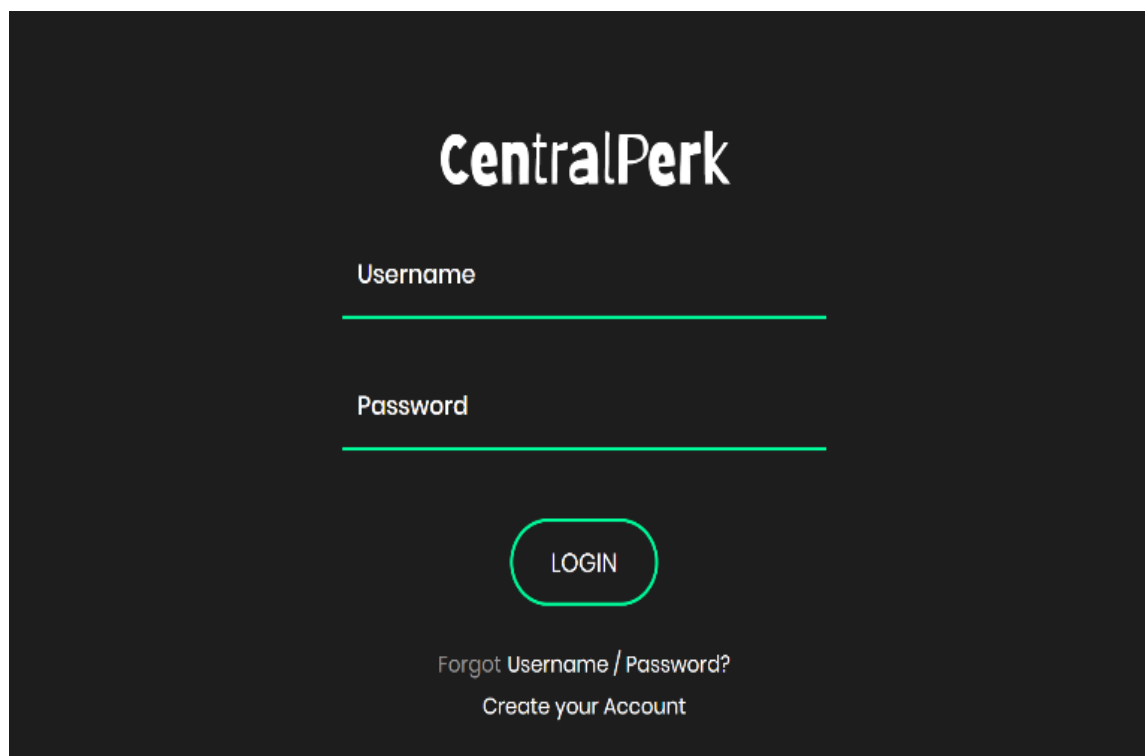


Fig 5.1: Login Page

The Fig 5.1 shows the login page of the application. When the login button is clicked, an AJAX POST request is sent to the server with the login credentials, the server checks the credentials and on success, the user gets logged-in and is redirected to the home page, else incorrect username or password message is displayed to the user. The “Create your Account” link at the bottom redirects user to the registration page. The AJAX request section is written using jQuery. Furthermore, a background task is scheduled on successful login, where any duplicate or existing sessions of the newly logged-in user is deleted from the system.

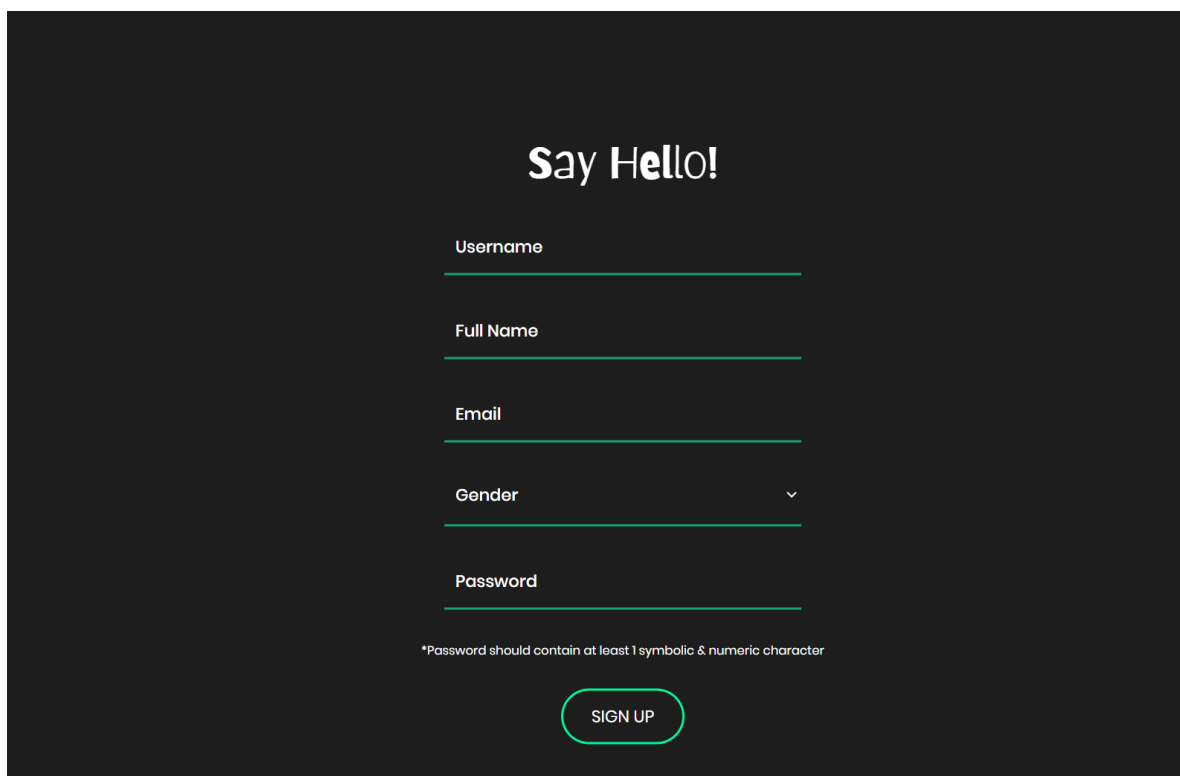
The image shows a registration form titled "Say Hello!" on a dark background. The form contains five input fields: "Username", "Full Name", "Email", "Gender" (a dropdown menu), and "Password". Each field has a light blue underline. Below the "Password" field, there is a small note: "*Password should contain at least 1 symbolic & numeric character". At the bottom of the form is a rounded rectangular button with the text "SIGN UP" in white.

Fig 5.2: Registration Page

The Fig 5.2 shows the signup page of the application. Each of the fields are server-side validated in real-time using AJAX. Celery is used to validate these fields, for example, the username is validated as if it is unique or are there any other user with the same username. Upon successful sign-up, the page gets redirected to home page, and upon unsuccessful sign-up, if there's any kind of validation error in any of the sign-

up fields, it gets shown under the field and the field's underline border changes its colour to red.

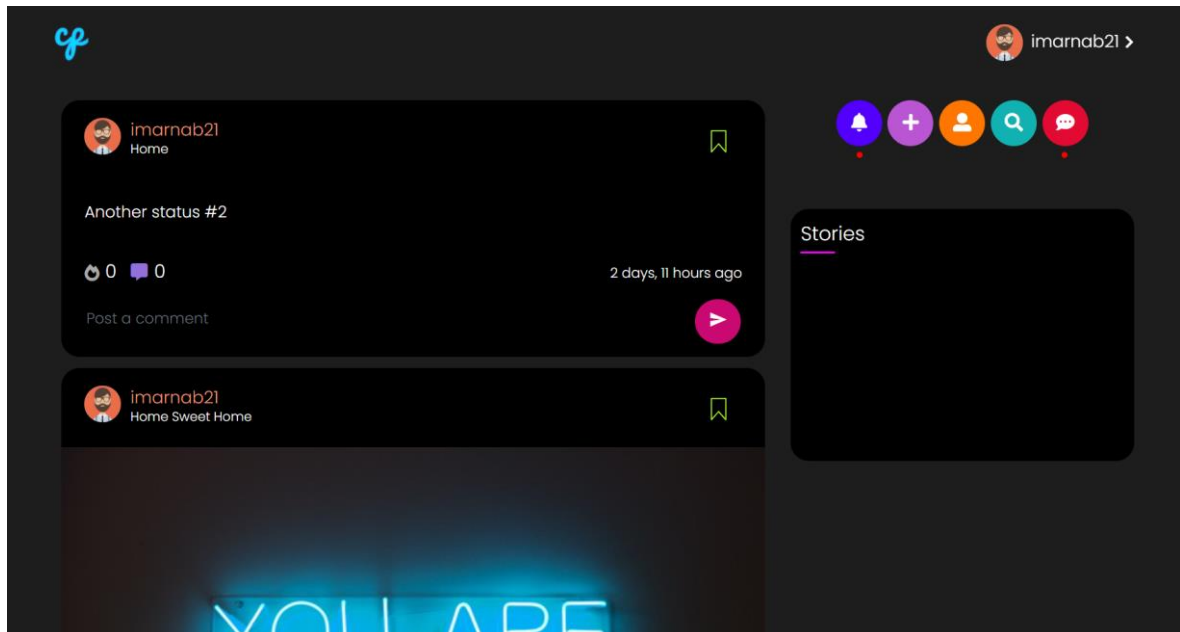


Fig 5.3: Home Page

The Figure 5.3 shows the home page of the application. This page is connected to the server through WebSocket connectivity. The left side of the page shows the post feed wall in the Home page of this application; All the user's and user's friend's recent posts are shown there. The right side of the page shows all the Modal buttons on the right side of the page, the first modal is the notification tab, second one is the friends tab where all the user's following and followers are shown, the third one is the post tab where the user can post pictures or status, the fourth one is the search tab used to search for an user by their username, and the final one is the messaging tab. The top rightmost section has the link to user's profile section and the logout button. All these tabs have automatic refresh feature which only gets refreshed from the server side.

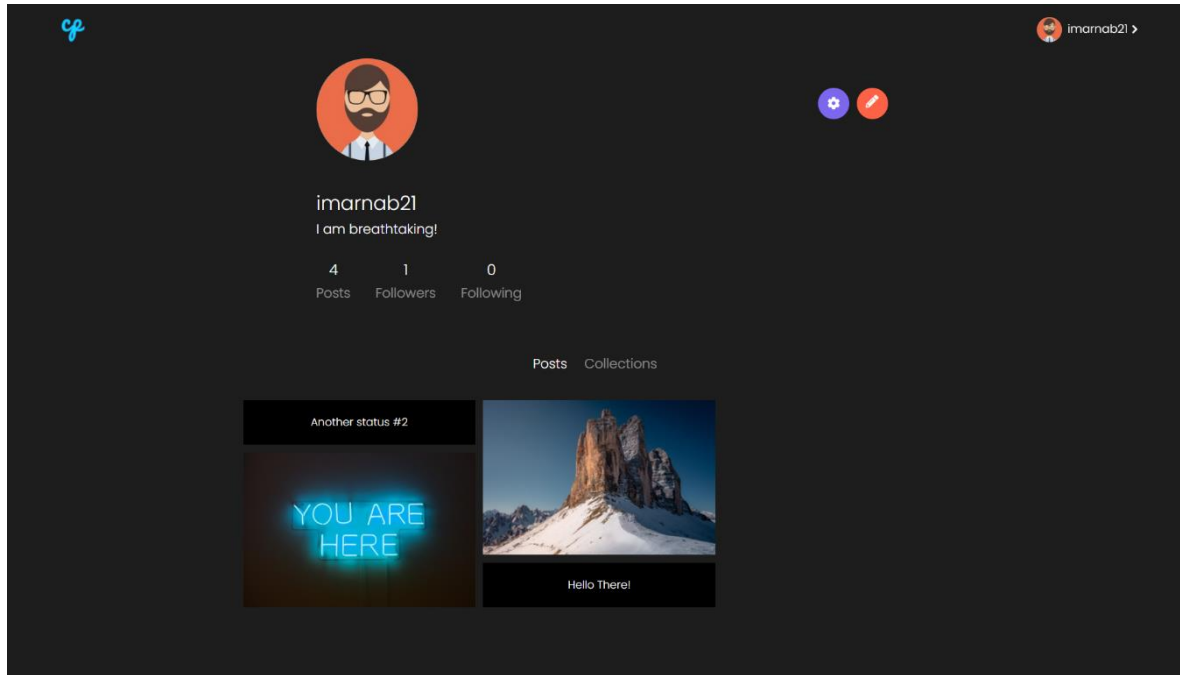


Fig 5.4: Profile Page

The Fig 5.4 shows the profile page of the application. The rightmost 2 buttons are the account settings modal and the edit profile modal, the first image is user's profile picture, then comes the user's username, then user's bio, then comes the count of user's post, followers and following. The bottom section of the page consists of user's post gallery and the posts that user has bookmarked. The posts in the post gallery is clickable and on-click, the page will be redirected to posts page, where the post details and like, comments will be shown. The thumbnails of post images are of lower resolution so that the page loads significantly faster, this lower resolution post image gets converted to and saved in the database during post upload.

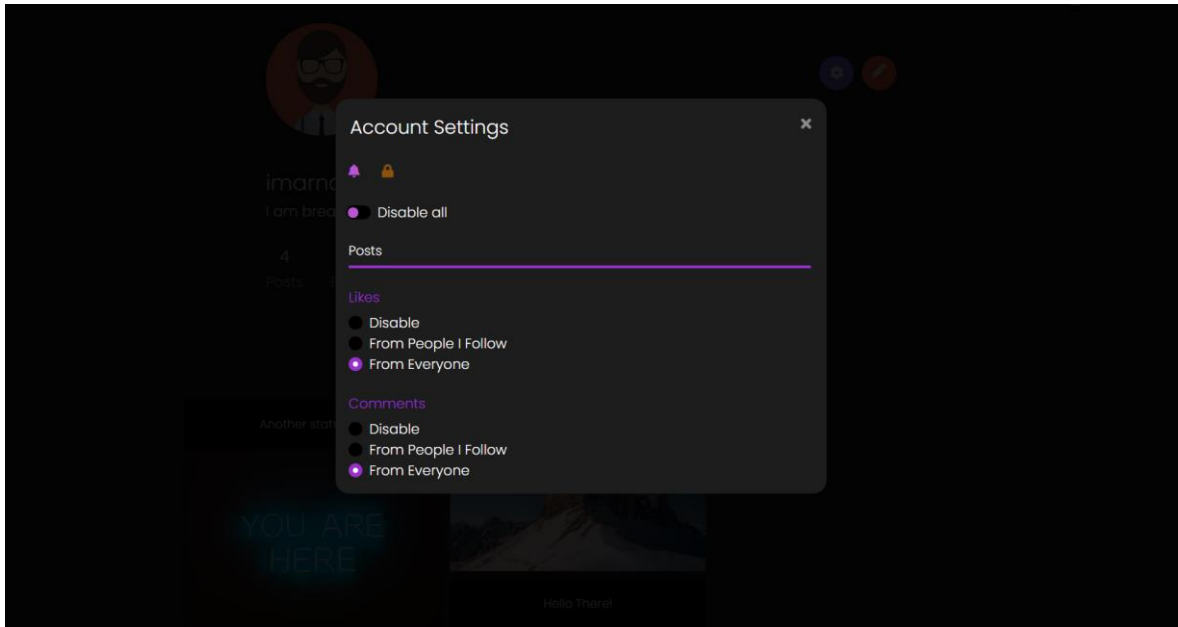


Fig 5.4.1: Account Settings section in Profile Page

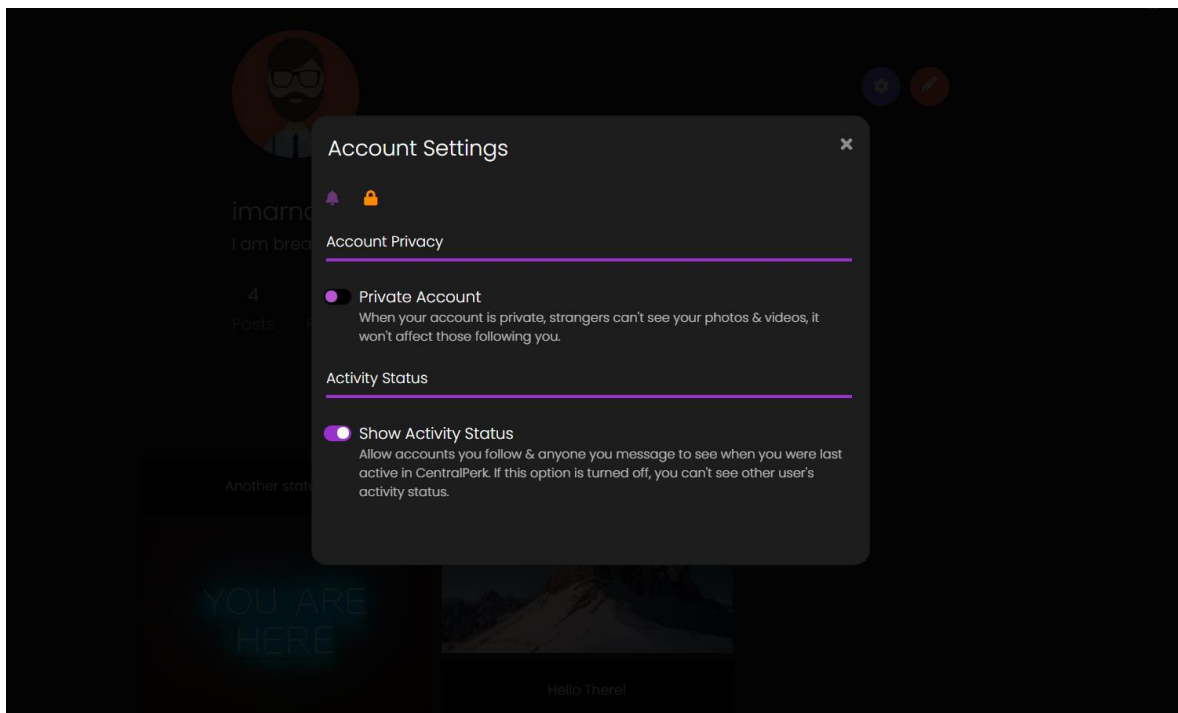


Fig 5.4.2: Account Privacy Settings section in Profile Page

Figure 5.4.1 and 5.4.2 shows the account privacy settings of the application. This section is made as minimal as possible. This includes disabling or enabling notifications of post likes, comments and comment likes, follower notifications, account privacy including enabling or disabling private account option.

cf

imamab21 >

Edit Profile

Edit Info Change Password



imamab21

Souma Kanti Ghosh

ghoshsoumakanti@gmail.com

16-06-1999

Male

I am breathtaking!

UPDATE

Fig 5.5: Edit Profile Page

The Fig 5.5 shows the edit profile page of the application. This includes changing profile photo, username, full name, email, birthdate, gender, bio as well as changing password. On clicking the update button, an AJAX POST request is sent to the server for validation and update, on success a notification is sent in real-time that the profile is successfully updated. The username as well as the email in this application must be unique, the user bio has 500 character word limit, and the user profile photo must be of jpeg and png file format, these essentials requirements gets checked off during the submitted form validation in the server side.

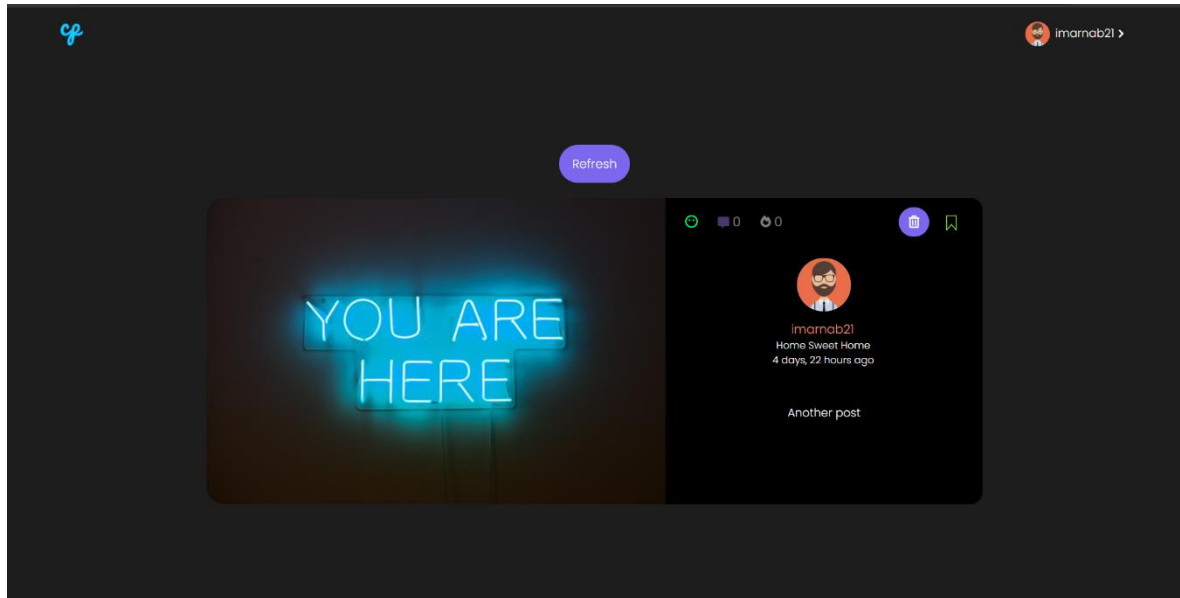


Fig 5.6.1: User post with image in Post page

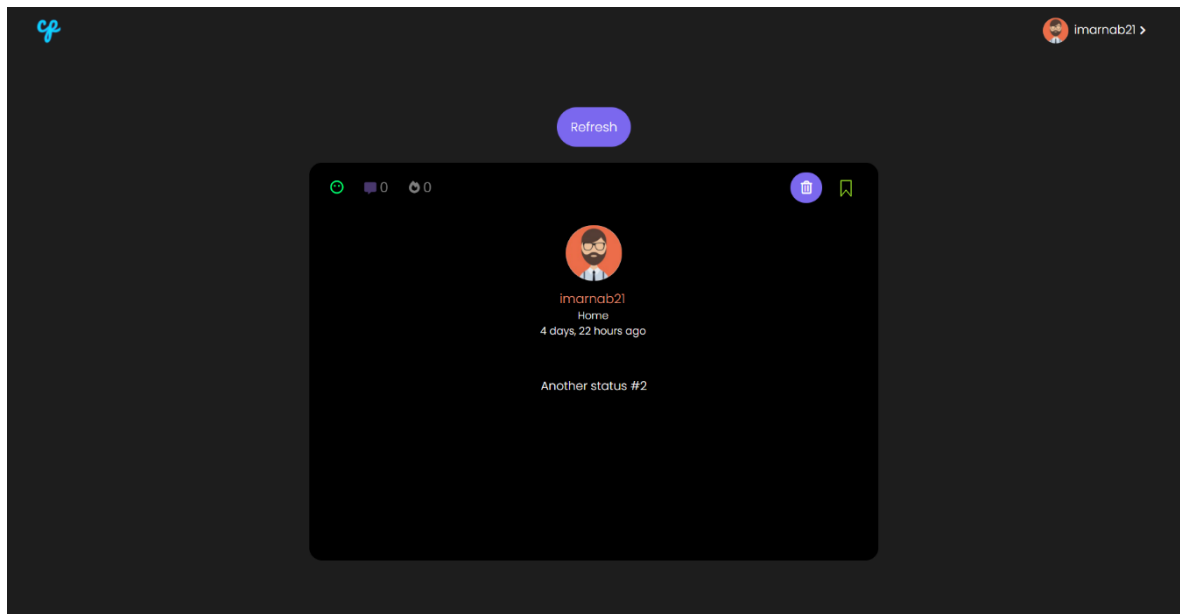


Fig 5.6.2: User status post in Post page

Figure 5.6.1 & 5.6.2 shows the post page of the application. The topmost button is the refresh button to refresh the contents in real-time using AJAX request/response being sent to/from the server. The left side of the tab contains the post photo, or blank in case of only status (Figure 5.6.2). The right side contains the user details who posted this image or status along with captions, comment section and like section. If the user is in his/her own post page, a delete option gets popped up on

the right side. There's a bookmark option in the rightmost corner of the post tab, which can be used to save a post in the user's collections gallery.

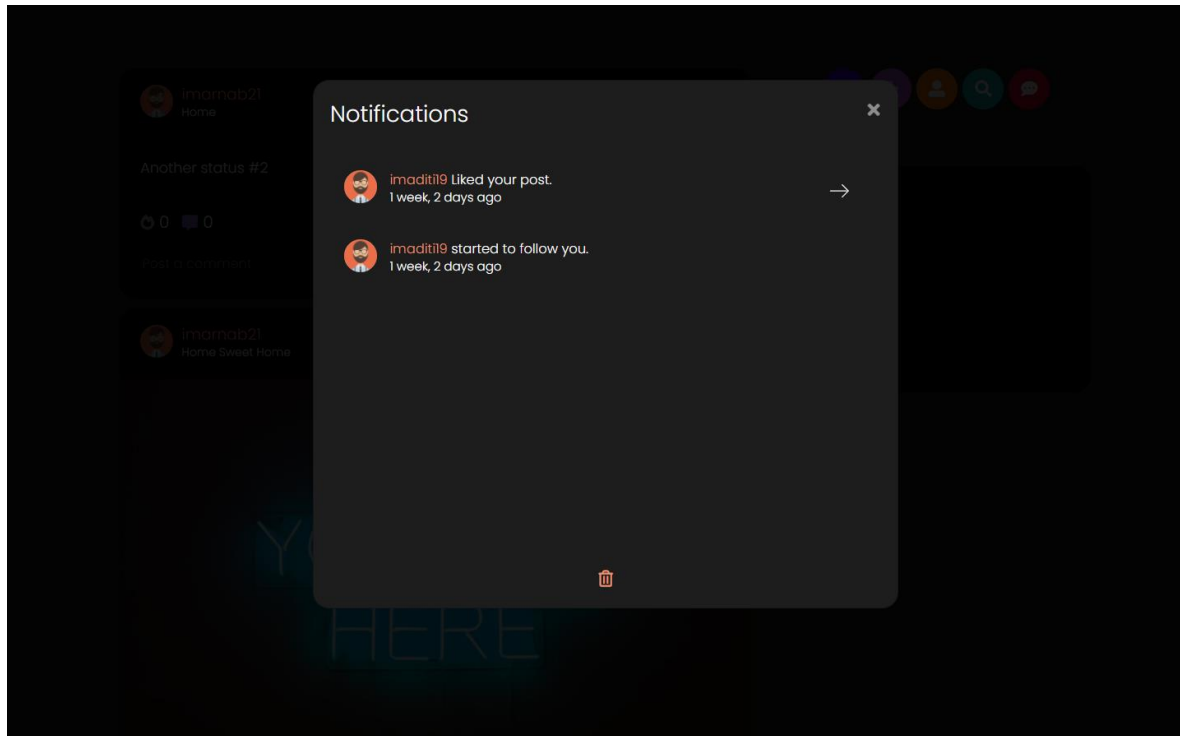


Fig 5.7: Notifications tab

The Fig 5.7 shows the notification tab in the home page of the application. Here all the types of notifications are shown to the user. As this page is connected to the server through WebSocket connectivity, this tab gets updated on real-time. The bottom button in the tab is used to clear all the notifications. This tab utilises bootstrap v4's modal feature and gets overlaid onto the user's screen as soon as the notifications modal button is clicked. Whenever new notifications gets pushed into this tab, a red dot is shown under the notifications modal button to notify the user that new content is available, same goes for every modal button in the home page.

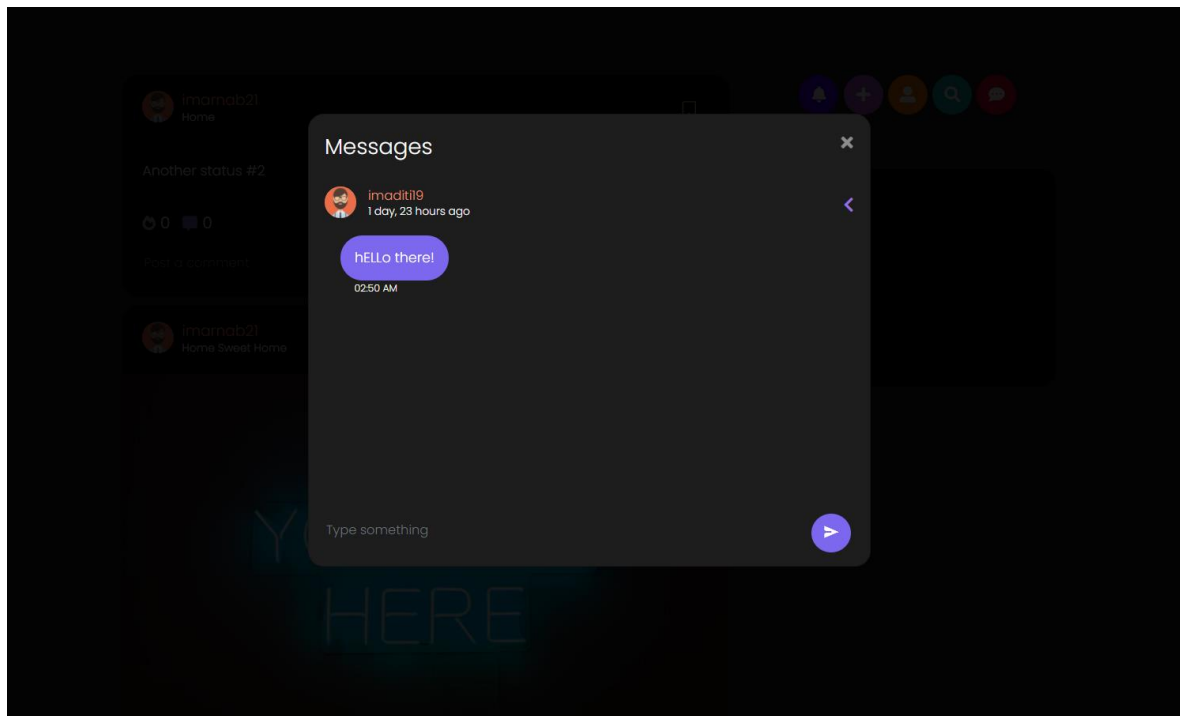


Fig 5.8: Messaging tab

The Fig 5.8 shows the messaging tab in the home page of the application. Here all the conversations are shown, the user can only send or receive messages from users he/she is friends with. The user can chat with only 1 user at a time. Unseen messages are saved in the database until it gets read, then it gets deleted from the server. If the user is online, a green dot is shown instead of the date & time at which the user was online, A typing notifier is also shown when the user is typing something to send to the opposite end. Message seen feature is also available in this application, an emoji with 2 eyes gets popped under the message last seen by the user at the opposite end. This tab utilises bootstrap v4's modal feature and gets overlaid onto the user's screen as soon as the message modal button is clicked.

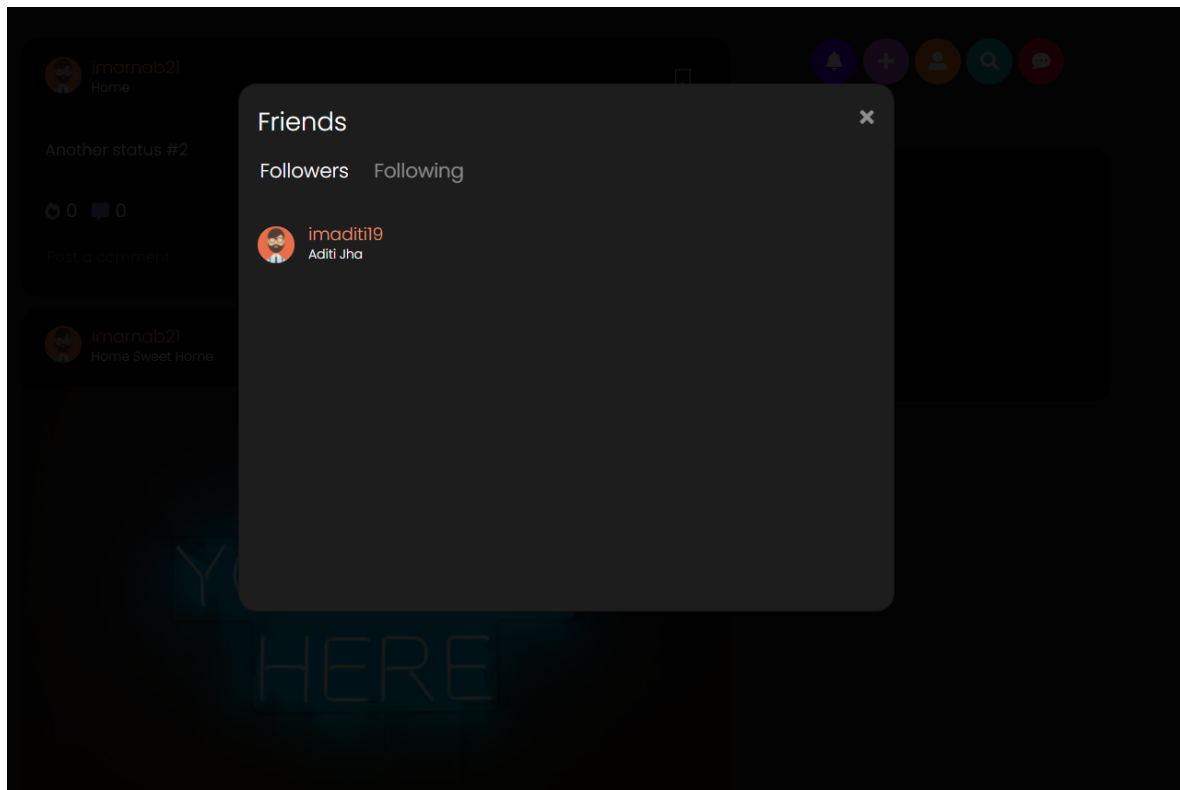


Fig 5.9: Friends tab

The Fig 5.9 shows the friends tab in the home page of the application. In this tab, the user followed or being followed by the logged-in user gets displayed, and the users in this list being online is highlighted with a green dot on the right of their username. This tab utilises bootstrap v4's modal feature and gets overlaid onto the user's screen as soon as the friends modal button is clicked. This modal, like every other modals gets updated in real-time.

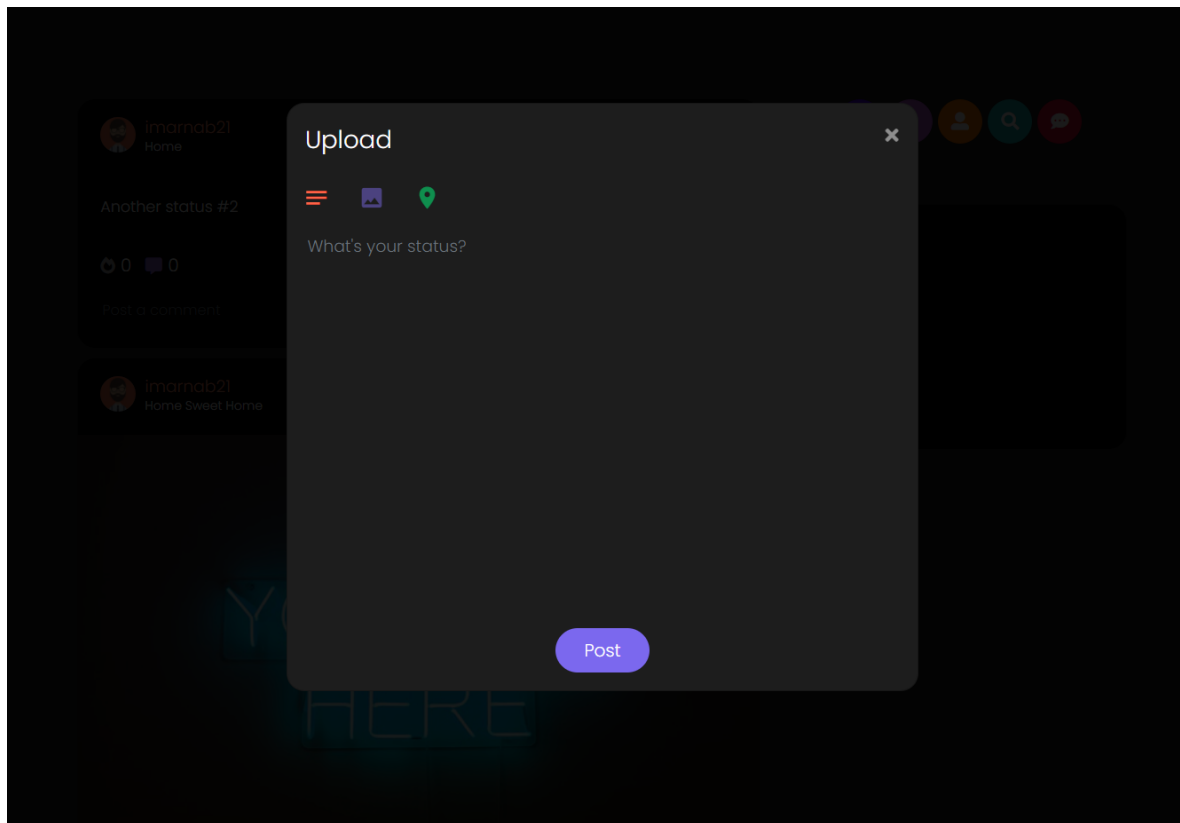


Fig 5.10: Upload tab

Figure 5.10 shows the Upload section in the home page of the application. In this tab, the user gets the opportunity to upload status or pictures with caption and location and post it on their feed. On click to post button, a POST request gets sent to the server, server-side validation including image format checks etc., are checked and on success the page gets refreshed with new post added on user's feed. When uploading an image for the post, it gets previewed in the upload panel. This tab utilises bootstrap v4's modal feature and gets overlaid onto the user's screen as soon as the notifications modal button is clicked.

CHAPTER 6

CONCLUSION AND FUTURE WORK

CONCLUSION

In the field of social media, the use of Django and the efficiency of WebSocket connectivity have brought a huge amount of benefits to both the server and the client. Applications like Instagram, utilising the power of Django, have been a boom in the social media industry. With the idea proposed in this paper, this project will minify the amount of privacy settings needed by the user, have unmonitored messaging service and a clean, responsive and attractive user interface, which will provide a seamless experience to the user. This system can be further extended with the additions of 24-hour stories, private stories, mentioning-user, toggling of chat privacy to make the application save or un-save (default) chats in the messaging section.

FUTURE WORK

In future, This Web Application can be one of the biggest social media sites in the market, but it still needs mammoth changes, including features like 24-hour stories, private stories, group chat communication, sharing of posts, mentioning user, toggling chat privacy to save chats in the messaging section and a lot more. A livestream or live video can also be added in this application, even though there are many features that are yet to be added in future, the layout of this application is very well formed and is scalable.

REFERENCES

- [1] Abhishek Bera, Shubham Darda, Ashish Gaikwad, Shivam Chanage and Prof. Suresh Rathod, "Professional Networking using Django.", International Journal of Advance Research in Computer Science and Software Engineering, IJARCSSE 6, no. 5 (2016).

- [2] Adama Shyam and Nitin Mukesh. "A Django based Educational Resource Sharing Website" *Journal of Specific Research*, JSR 64, no. 1 (2020).
- [3] M. Lunacek, J. Braden and T. Hauser, "The scaling of many-task computing approaches in python on cluster supercomputers," *2013 IEEE International Conference on Cluster Computing (CLUSTER)*, Indianapolis, IN, USA, 2013, pp. 1-8, doi: 10.1109/CLUSTER.2013.6702678.
- [4] P. Pawlak, B. Sakowicz, P. Mazur and A. Napieralski, "Social network application based on Google Web Toolkit," *2009 10th International Conference - The Experience of Designing and Application of CAD Systems in Microelectronics*, Lviv, Ukraine, 2009, pp. 461-464.
- [5] J. Kiruthika, S. Khaddaj, D. Greenhill and J. Francik, "User Experience Design in Web Applications," *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, Paris, 2016, pp. 642-646, doi: 10.1109/CSE-EUC-DCABES.2016.253.
- [6] A. Pinandito, H. M. Az-zahra, L. Fanani and A. V. Putri, "Analysis of web content delivery effectiveness and efficiency in responsive web design using material design guidelines and User Centered Design," *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, Malang, Indonesia, 2017, pp. 435-441, doi: 10.1109/SIET.2017.8304178.
- [7] Lijing Zhang and Xiaoxiao Shen, "Research and development of real-time monitoring system based on WebSocket technology," *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, Shenyang, China, 2013, pp. 1955-1958, doi: 10.1109/MEC.2013.6885373.
- [8] S. Li, H. Jiang and M. Shi, "Redis-based web server cluster session maintaining technology," *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, Guilin, China, 2017, pp. 3065-3069, doi: 10.1109/FSKD.2017.8393274.
- [9] Django, A Python Web framework encouraging rapid development and clean, pragmatic design, at <https://www.djangoproject.com/>.

- [10] Django Channels, a project that takes Django and extends its abilities beyond HTTP - to handle WebSockets, referenced at <https://channels.readthedocs.io/en/stable/>.
- [11] Bootstrap, a free and open-source CSS framework directed at responsive, mobile-first front-end web development, referenced at <https://getbootstrap.com/>.

APPENDIX

A. Paper Acceptance Mail

B. Plagiarism Report

Social Network Web Application

Souma Kanti Ghosh
UG scholar

Department of Computer Science
Sathyabama Institute of Science
and Technology, Chennai,
Tamil Nadu.
ghoshsoumakanti@gmail.com

Abhishek Kumar
UG scholar

Department of Computer Science
Sathyabama Institute of Science
and Technology, Chennai,
Tamil Nadu.
abhishek21424@gmail.com

Dr. S.Revathy

Associate professor
School of Computing
Sathyabama Institute of Science
and Technology, Chennai,
Tamil Nadu.
ramesh.revathy@gmail.com

Abstract

In recent years, social media have attracted considerable attention. Social media giants like Facebook, Instagram etc., have over billions of users using their applications, but most of these applications in the world wide web uses user's privacy to monitor, analyse and sell their private data. Moreover, some of the apps have complex privacy settings, not even needed by the user but are enabled by default. In this paper, a full-fledged Social Network Web Application is proposed. The main objective of the paper is to solve the issue of too many privacy settings, monitored messaging service and complex user interface (UI) in social network web apps nowadays. Users can use this application to create and customize their own profile, upload and share pictures or status with their friends, which can be liked and commented upon, notification system, with the addition of an easy-to-use UI, lesser privacy settings and unmonitored messaging service to ease the user experience.

Keywords:- Web Application, Social Network, User Experience, User Interface, messaging service.

I. INTRODUCTION

In this era of digitalization, Social Networking is the use of Internet-based social media sites to stay connected with friends, family, colleagues, customers or client. Social Networking can have a social purpose, a business purpose, or both, through sites like Facebook, Instagram, among others. This web application is based on the idea of a full-fledged social connectivity network similar to that of Facebook, and it aims to solve the issue of too many privacy settings, monitored messaging service and complex user interface (UI) in social media sites nowadays. This application is developed using Django, Python in backend and Bootstrap v4, jQuery in frontend, and is delivered as a Web Application.

II. RELATED WORK

Abhishek Bera, et al [1] developed a decentralized professional networking application using Django, an online service in which users can sign-up, search and access contents

relevant to their knowledge, as well as connect with users of similar interest to gain experience, learn and develop their skills.

Adamy Shyam, et al [2] developed an Educational Resource Sharing Web Application using Django, with the help of which students would access different types of educational books or documents and sell them as well in the same platform.

M. Lunacek et al [3] compared two packages in Python designed especially for performing Many Task Computing (MTC), IPython-Parallel and Celery, to study scaling using over 12000 cores and several thousand tasks, using mpi4py as a baseline for the comparisons.

P. Pawlak et al [4] developed a social networking application which is based on Google Web Toolkit (GWT), a technology in creation of rich AJAX based applications and using Java as the programming language, which is compiled into pure JavaScript and deployed as a Web Application.

J. Kiruthika et al [5] studied the issues related to web application designs leading to dissatisfaction of users and also recommended some of the design criteria to create efficient web applications.

A. Pinandito et al [6] compared and analysed the effectiveness and efficiency of a web page being displayed on a computer screen, tablets and smartphones and also shows how User Centred Design and Material Design guidelines improve the efficiency of content delivery in web pages on devices of different screen size.

Lijing Zhang et al [7] developed a real-time monitoring system based on WebSocket connectivity and also analysed the traditional model of Web applications and the work mode of the push feature in WebSocket technology.

S. Li et al [8] analysed several commonly used Web server cluster session maintaining methods and developed a Redis-based Web server cluster to maintain several sessions to make the cluster scale out more easily and efficiently.

III. PROPOSED WORK

This paper provides a web based social network application developed using a Python based backend framework, Django along with Channels, which extends Django’s abilities to handle other connections in either a synchronous or asynchronous style. In this paper, A Python based framework “Celery” is being used to handle any sort of background task handling like saving unseen messages in the database, sending notifications to a user etc., Along with that, PostgreSQL is used for data storage, Redis is used for caching and also to be a message broker. In the frontend section, the UI is developed to be as clean as possible, with minimal animations, using Bootstrap as a frontend framework and utilising its overlay feature – “Modals” for different sections in the application, from notifications panel to user messaging section.

The main objective of the paper is to implement a fast & efficient, social network web application with minimal privacy settings, unmonitored messaging service and a clean, lightweight UI. The main page or the home page of the application which opens up right after signing in is connected to the server through WebSocket connection, handled by Channels, which makes asynchronous requests between server and client, messaging service possible. Other essential features of this application consist of uploading status or pictures, liking and commenting on posts, bookmarking posts, private and public account settings, accepting or declining friend requests, personal profile page with posts and bookmarks gallery, edit profile section, changing password section as well.

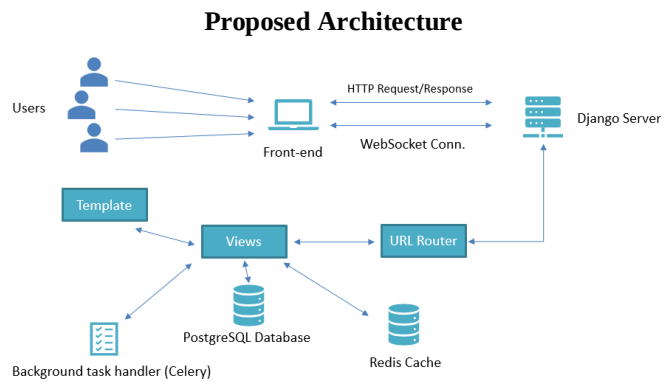


Fig.1. The architecture of proposed work

Fig 1 represents the architecture of the proposed work. Here the users interact with the frontend of the application, any type of request from the client is sent to the server for a response. The server accepts two type of connection requests, HTTP and WebSocket connectivity. The server side is developed using Django with Channels extension to handle WebSocket connections, which sends a response to the client based on the request. For any kind of background task handling, the server

side connects with Celery workers, for data storage the server connects with PostgreSQL Database used for this application and for caching purposes, in-memory cache Redis is used. Redis is also used by Channels and celery as a message broker.

BACKEND FRAMEWORKS AND PACKAGES

A. DJANGO SERVER

Django is a web application framework written in Python programming language. It is based on MVT (Model View Template) design pattern. The Django is very demanding due to its rapid development feature. It takes less time to build application after collecting client requirement. Django was design and developed by Lawrence journal world in 2003 and publicly released under BSD license in July 2005. Currently, DSF (Django Software Foundation) maintains its development and release cycle. The Template is a presentation layer which handles User Interface part completely. The View is used to execute the business logic and interact with a model to carry data and renders a template.

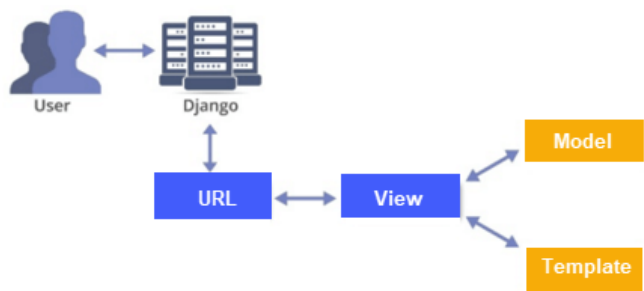


Fig.2. Django MVT Structure

B. DJANGO CHANNELS

Channels is a project that takes Django and extends its abilities beyond HTTP – to handle WebSocket, chat protocols, IoT protocols and more. It’s built on a Python specification called ASGI. Channels builds upon the native ASGI support available in Django since v3.0, and provides an implementation itself for Django v2.2. Django still handles traditional HTTP, whilst Channels give the choice to handle other connections in either a synchronous or asynchronous style.

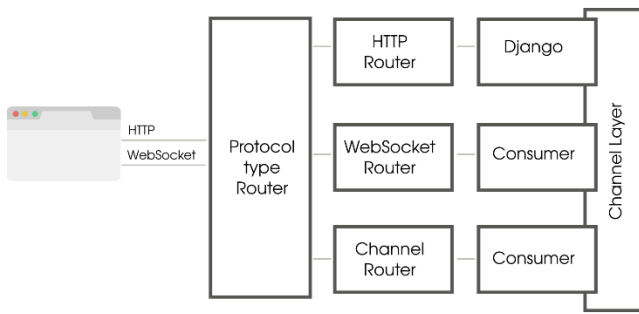


Fig.3. Channels structure over Django

A. CELERY

Celery is a simple, flexible, and reliable distributed system to process vast amounts of messages, while providing operations with the tools required to maintain such a system. It's a task queue with focus on real-time processing, while also supporting task scheduling.

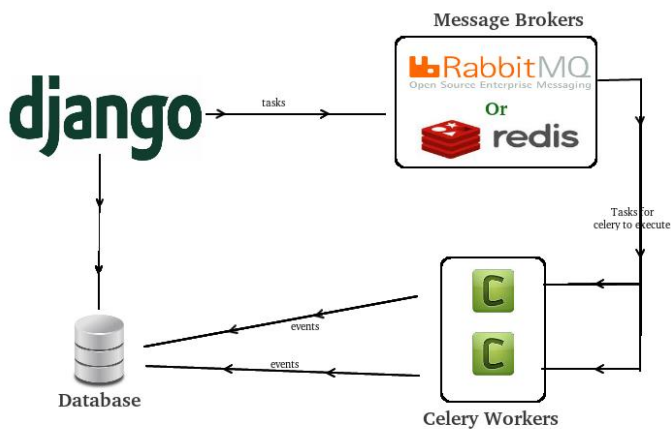


Fig.4. Django with Celery integration structure

DATA STORAGE

A. POSTGRESQL DATABASE

PostgreSQL, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance. It was originally named POSTGRES, referring to its origins as a successor to the Ingres database developed at the University of California, Berkeley. It is a highly stable database management system, backed by more than 20 years of community development which has contributed to its high levels of resilience, integrity, and correctness. **PostgreSQL is used as the primary data store or data warehouse for many web, mobile, geospatial, and analytics applications.**

CACHING STORAGE AND MESSAGE BROKER

A. REDIS

Redis is an in-memory data structure store, used as a distributed, in-memory key-value database, cache and message broker, with optional durability. Redis supports

different kinds of abstract data structures, such as strings, lists, maps, sets, sorted sets, HyperLogLogs, bitmaps, streams, and spatial indexes.

FRONTEND FRAMEWORKS AND PACKAGES

A. BOOTSTRAP

Bootstrap is one of the most popular front-end open source toolkits, featuring Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful JavaScript plugins. Version 4 of Bootstrap had jQuery, which was removed in the latest version (5.0).

B. JQUERY

jQuery is a JavaScript library designed to simplify HTML DOM tree traversal and manipulation, as well as event handling, CSS animation, and AJAX. It is free, open-source software using the permissive MIT license. As of May 2019, jQuery is used by 73% of the 10 million most popular websites.

C. CSS

Cascading Style Sheets is a style sheet language used for describing the presentation of a document written in a mark-up language such as HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. This language contains coding elements and is composed of these "cascading style sheets" which are equally called CSS files.

IV. RESULT AND DISCUSSION

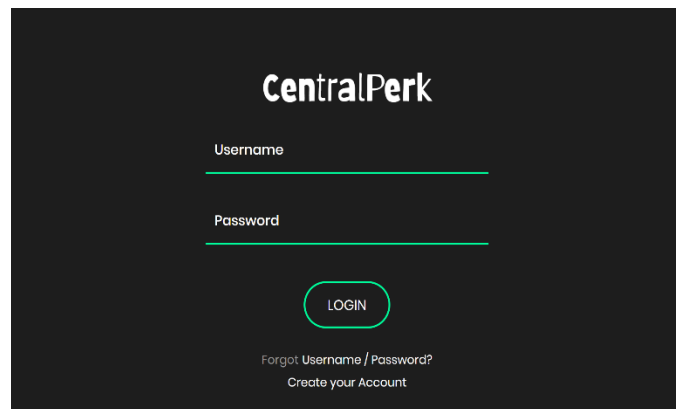


Fig.5. Login Page

Figure 5 shows the login page of the application. When the login button is clicked, an AJAX POST request is sent to the server with the login credentials, the server checks the credentials and on success, the user gets logged-in and is redirected to the home page, else incorrect username or password message is displayed to the user. Furthermore, a background task is scheduled on successful login, where any duplicate or existing sessions of the newly logged-in user is deleted from the system.

Say Hello!

Username

Full Name

Email

Gender

Password

*Password should contain at least 1 symbolic & numeric character

SIGN UP

Fig.6. Signup Section

Figure 6 shows the signup page of the application. Each of the fields are server-side validated in real-time using AJAX. Celery is used to validate these fields, for example, the username is validated as if it is unique or are there any other user with the same username. Upon successful sign-up, the page gets redirected to home page, and upon unsuccessful sign-up, if there's any kind of validation error in any of the sign-up fields, it gets shown under the field and the field's underline border changes its colour to red.

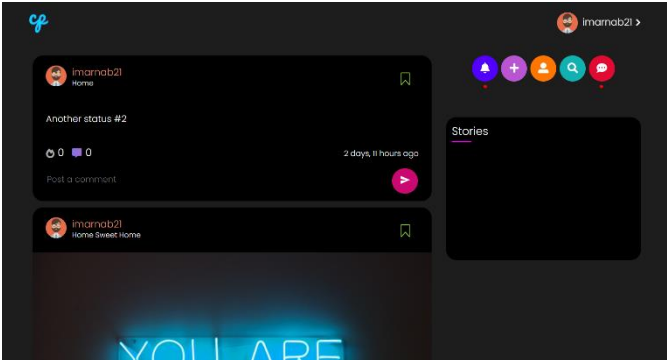


Fig.7. Home page

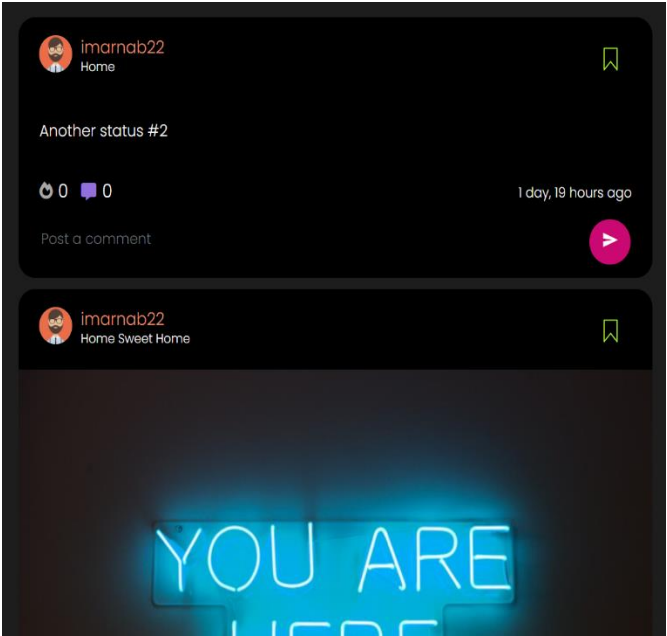


Fig.7.1. User post feed section in Home page

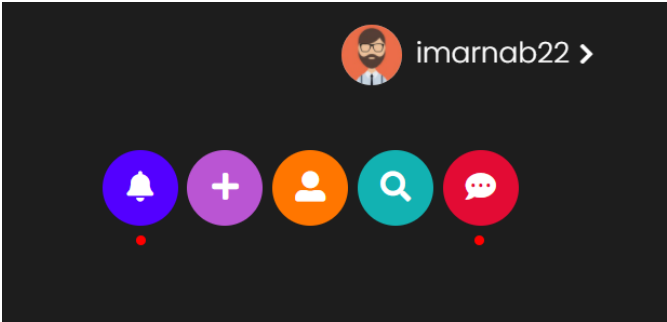


Fig.7.2. Modal buttons for different tabs in Home page

Figure 7 shows the home page of the application. This page is connected to the server through WebSocket connectivity. Figure 7.1 shows the post feed wall in the Home page of this application; All the user's and user's friend's recent posts are

shown there. Figure 7.2 shows all the Modal buttons on the right side of the page, the first modal is the notification tab, second one is the friends tab where all the user’s following and followers are shown, the third one is the post tab where the user can post pictures or status, the fourth one is the search tab used to search for an user by their username, and the final one is the messaging tab. The top rightmost section has the link to user’s profile section and the logout button.

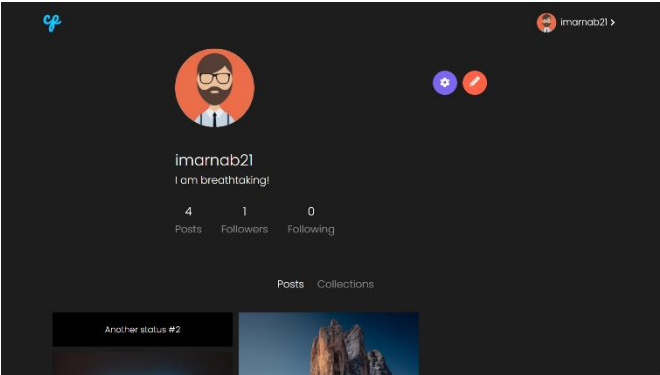


Fig.8. Profile page

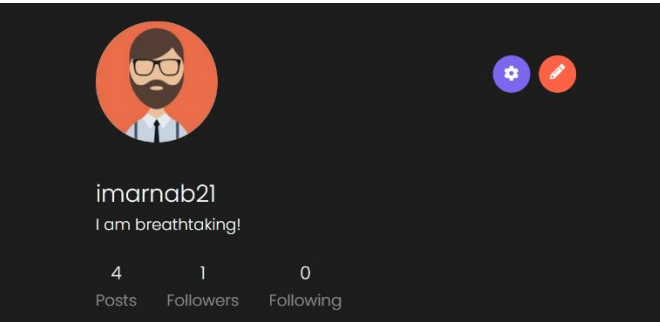


Fig.8.1. User Info section in Profile page

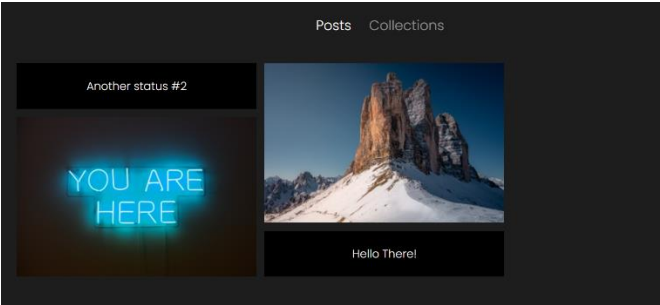


Fig.8.2. User post gallery section in Profile page

Figure 8 shows the profile page of the application. Figure 8.1 shows the rightmost 2 buttons which are the account settings modal and the edit profile modal, the first image is user’s profile picture, then comes the user’s username, then user’s bio, then comes the count of user’s post, followers and following. Figure 8.2 shows the bottom section of the page which consists of user’s post gallery and the posts that user has bookmarked.

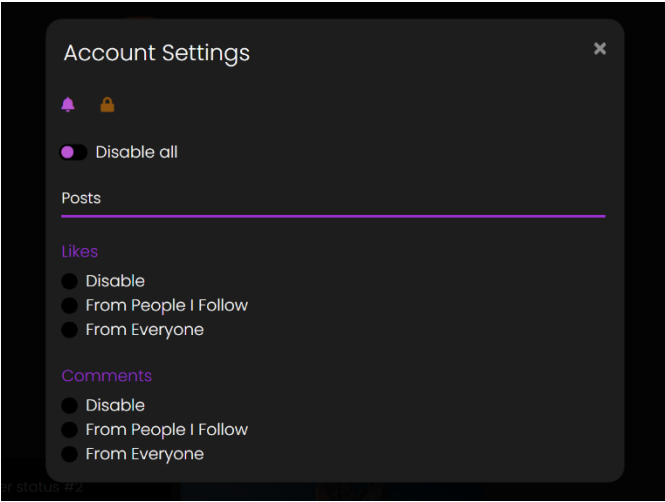


Fig.9. Account privacy settings

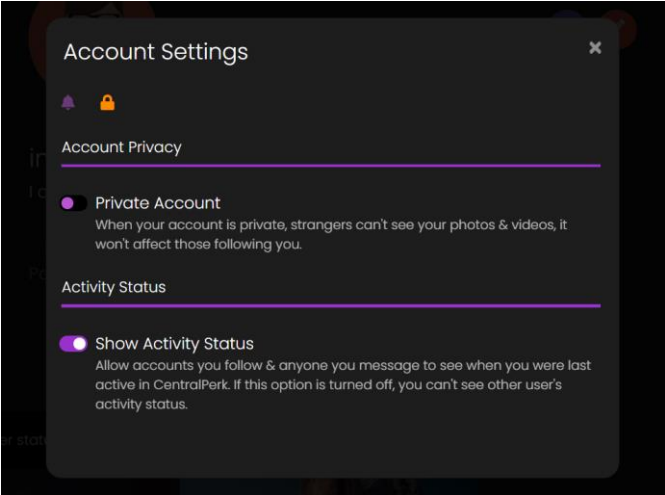


Fig.9.1. Account privacy settings

Figure 9 and 9.1 shows the account privacy settings of the application. This section is made as minimal as possible. This includes disabling or enabling notifications of post likes, comments and comment likes, follower notifications, account privacy including enabling or disabling private account option.

Fig.10. Edit Profile section

Figure 10 shows the edit profile page of the application. This includes changing profile photo, username, full name, email, birthdate, gender, bio as well as changing password. On clicking the update button, an AJAX POST request is sent to the server for validation and update, on success a notification is sent in real-time that the profile is successfully updated.

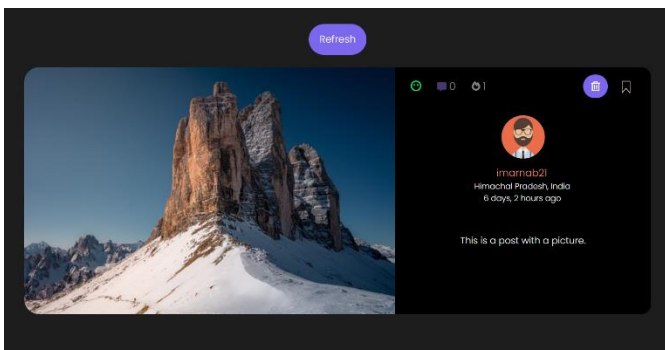


Fig.11.1. User post with image in post page

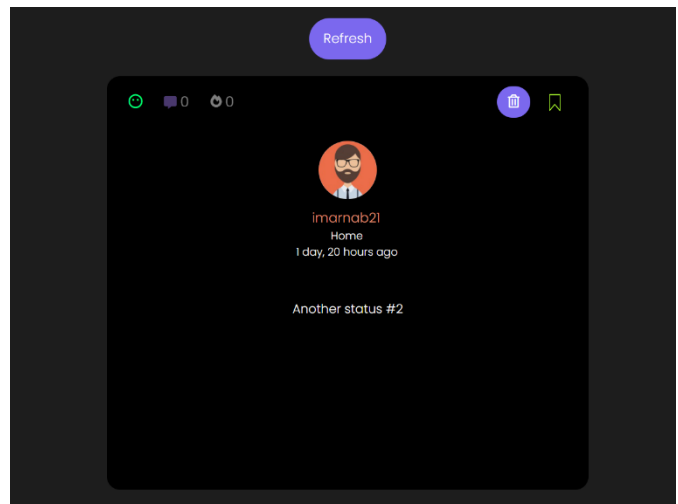


Fig.11.2. User status post in post page

Figure 11.1 & 11.2 shows the post page of the application. The topmost button is the refresh button to refresh the contents in real-time using AJAX request/response being sent to/from the server. The left side of the tab contains the post photo, or blank in case of only status (Figure 11.2). The right side contains the user details who posted this image or status along with captions, comment section and like section. If the user is in his/her own post page, a delete option gets popped up on the right side.

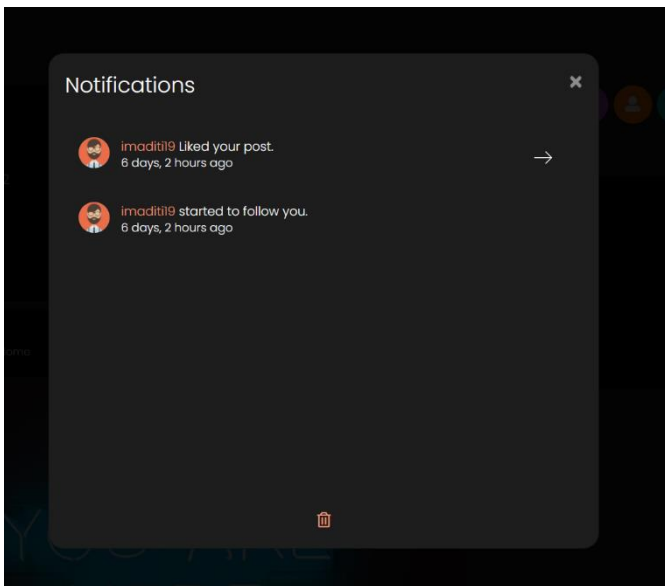


Fig.12. Notifications tab in Home page

Figure 12 shows the notification tab in the home page of the application. Here all the types of notifications are shown to the user. As this page is connected to the server through WebSocket connectivity, this tab gets updated on real-time. The bottom button in the tab is used to clear all the notifications.

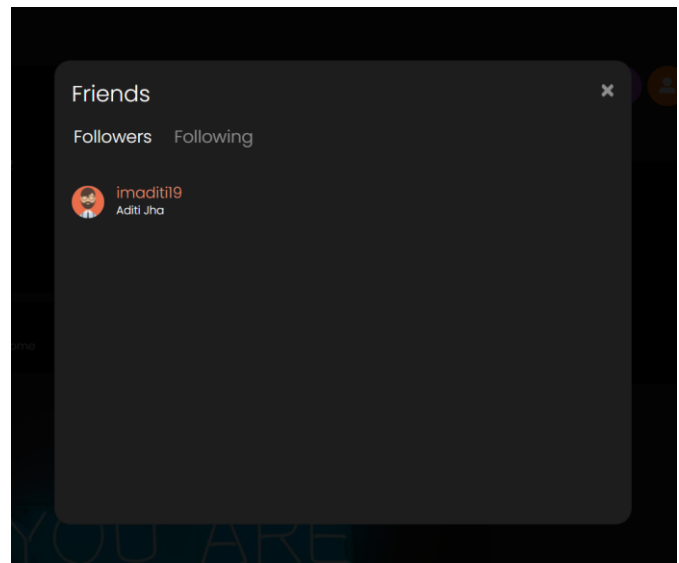


Fig.14. Friends tab in Home page

Figure 14 shows the friends tab in the home page of the application. In this tab, the user followed or being followed by the logged-in user gets displayed, and the users in this list being online is highlighted with a green dot on the right of their username.

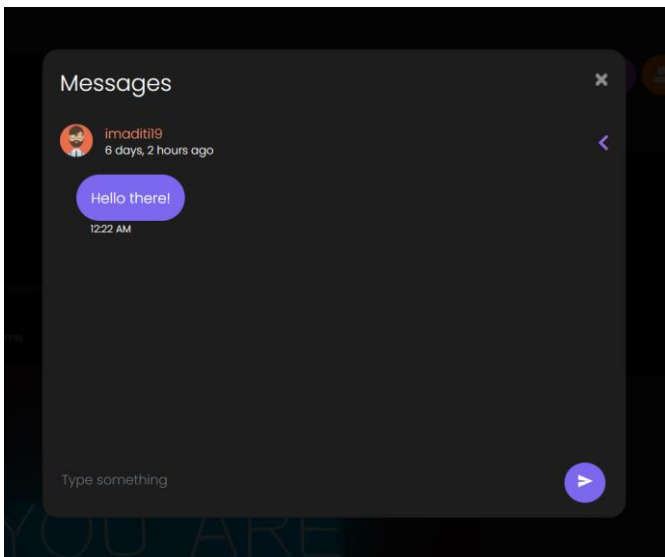


Fig.13. Messaging tab in Home page

Figure 13 shows the messaging tab in the home page of the application. Here all the conversations are shown, the user can only send or receive messages from users he/she is friends with. The user can chat with only 1 user at a time. Unseen messages are saved in the database until it gets read, then it gets deleted from the server.

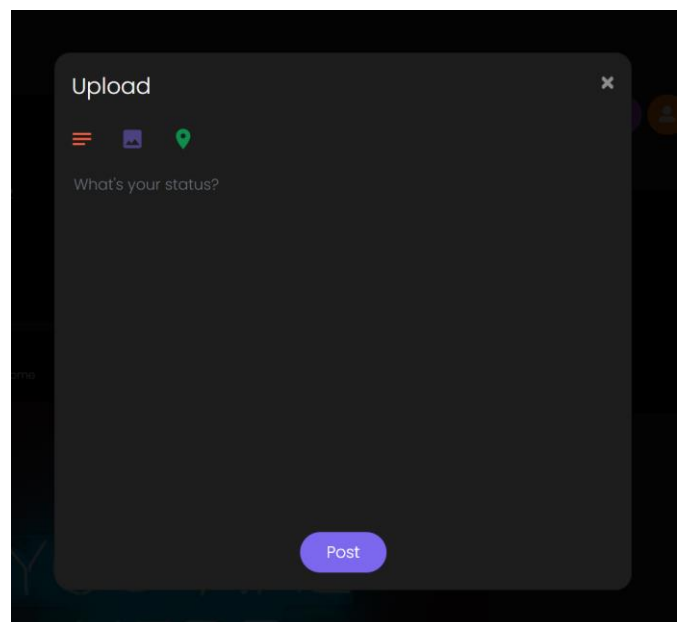


Fig.15. Upload tab in Home page

Figure 15 shows the Upload section in the home page of the application. In this tab, the user gets the opportunity to upload status or pictures with caption and location and post it on their feed. On click to post button, a POST request gets sent to the server, server-side validation including image format checks

etc., are checked and on success the page gets refreshed with new post added on user’s feed.

V. PERFORMANCE ANALYSIS

The performance of this web application was analysed using Google’s Lighthouse. Google Lighthouse is an open-source, automated tool for measuring the quality of web pages. It can be run against any web page, public or requiring authentication. Google Lighthouse audits performance, accessibility and search engine optimization of web pages.

Performance analysis of content heavy pages in this web application is being shown, these pages are the Home page and the Profile page which has most of the contents of all pages in the application, both mobile and desktop mode performances has been measured, Google Chrome browser has been used for this purpose and the pages are opened in incognito mode.

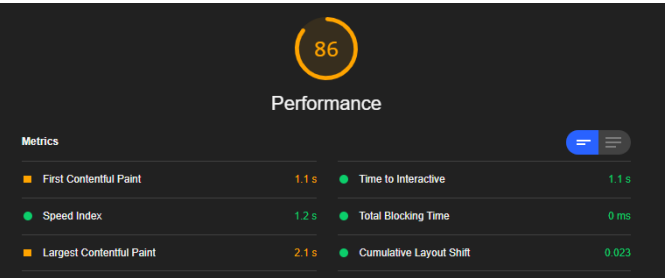


Fig.16. Home page performance (Desktop mode)

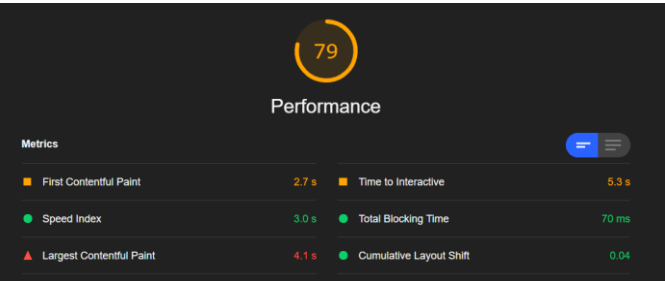


Fig.17. Home page performance (Mobile mode)

Figure 16 shows the performance analysis of the Home page in Desktop mode by Google Lighthouse. Figure 17 shows the performance analysis of the same page in Mobile mode, in which the responsiveness of this application under smaller screens has been tested.

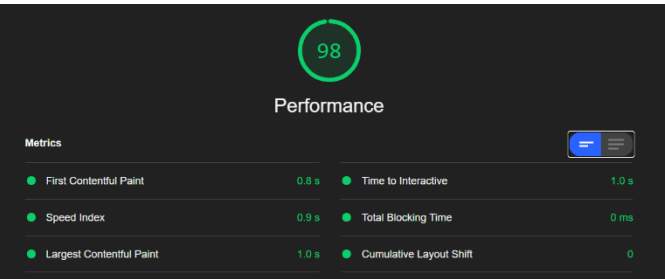


Fig.18. Profile page performance (Desktop mode)

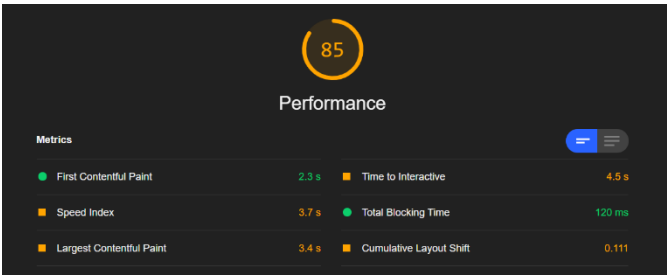


Fig.19. Profile page performance (Mobile mode)

Figure 18 shows the performance analysis of the Profile page in Desktop mode by Google Lighthouse, Figure 19 shows the performance analysis of the Profile page in Mobile mode.

VI. CONCLUSION

In the field of social media, the use of Django and the efficiency of WebSocket connectivity have bought a huge amount of benefits to both the server and the client. Applications like Facebook and Instagram have been a boom in the social media industry. With the idea proposed in this paper, this project will minify the amount of privacy settings needed by the user, have unmonitored messaging service and a clean, responsive and attractive user interface, which will provide a seamless experience to the user. This system can be further extended with the additions of 24-hour stories, private stories, mentioning-user, toggling of chat privacy to make the application save or un-save (default) chats in the messaging section.

VII. REFERENCES

[1] Abhishek Bera, Shubham Darda, Ashish Gaikwad, Shivam Chanage and Prof. Suresh Rathod, "Professional Networking using Django.", International Journal of Advance Research in Computer Science and Software Engineering, IJARCSSE 6, no. 5 (2016).

[2] Adama Shyam and Nitin Mukesh. "A Django based Educational Resource Sharing Website" Journal of Specific Research, JSR 64, no. 1 (2020).

M. Lunacek, J. Braden and T. Hauser, "The scaling of many-task computing approaches in python on cluster supercomputers," 2013 IEEE

- [1] *International Conference on Cluster Computing (CLUSTER)*, Indianapolis, IN, USA, 2013, pp. 1-8, doi: 10.1109/CLUSTER.2013.6702678.
- [2] P. Pawlak, B. Sakowicz, P. Mazur and A. Napieralski, "Social network application based on Google Web Toolkit," *2009 10th International Conference - The Experience of Designing and Application of CAD Systems in Microelectronics*, Lviv, Ukraine, 2009, pp. 461-464.
- [3] J. Kiruthika, S. Khaddaj, D. Greenhill and J. Francik, "User Experience Design in Web Applications," *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, Paris, 2016, pp. 642-646, doi: 10.1109/CSE-EUC-DCABES.2016.253.
- [4] A. Pinandito, H. M. Az-zahra, L. Fanani and A. V. Putri, "Analysis of web content delivery effectiveness and efficiency in responsive web design using material design guidelines and User Centered Design," *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, Malang, Indonesia, 2017, pp. 435-441, doi: 10.1109/SIET.2017.8304178.
- [5] Lijing Zhang and Xiaoxiao Shen, "Research and development of real-time monitoring system based on WebSocket technology," *Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)*, Shenyang, China, 2013, pp. 1955-1958, doi: 10.1109/MEC.2013.6885373.
- [6] S. Li, H. Jiang and M. Shi, "Redis-based web server cluster session maintaining technology," *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, Guilin, China, 2017, pp. 3065-3069, doi: 10.1109/FSKD.2017.8393274.

D. Source Code

```
from django.db import models
from django.contrib.auth.models import AbstractBaseUser
from django.db.models.signals import post_delete
from django.dispatch import receiver
from Profile.user_manager import UserManager
import uuid

# Create your models here.

GENDER = ( ('Male', 'Male'), ('Female', 'Female'), ('Other',
'Other'), )

def user_directory_path(instance, filename):
    # file will be uploaded to MEDIA_ROOT/<user_id>/<filename>
    return f"profile_pics/{instance.user_id}/{filename}"

class User(AbstractBaseUser):
    user_id = models.UUIDField(primary_key=True,
default=uuid.uuid4, editable=False)
    session_key = models.CharField(max_length=40,
default='notyetacquired')
    monitor_task_id = models.CharField(max_length=60,
blank=True, null=True)
    channel_name = models.CharField(max_length=100,
default="", blank=True, null=True)
    username = models.CharField(max_length=20,
default="", unique=True)
    full_name = models.CharField(max_length=50)
    birthdate = models.CharField(max_length=10)
    bio = models.TextField(max_length=160, default="I am
breathhtaking!", blank=True)
    gender = models.CharField(max_length=20,
choices=GENDER)
    email = models.EmailField(max_length=255, unique=True)
    created = models.DateTimeField(auto_now_add=True,
auto_now=False)
```

```

just_created = models.BooleanField(default=True)
profile_pic = models.ImageField(blank=True,
upload_to=user_directory_path)
admin = models.BooleanField(default=False) # a superuser
staff = models.BooleanField(default=False) # a admin user;
non super-user
active = models.BooleanField(default=True)

USERNAME_FIELD = 'username'
REQUIRED_FIELDS = ['email',]

def __str__(self):
    return self.username

def is_staff(self):
    return self.staff

def is_admin(self):
    return self.admin

def is_active(self):
    return self.active

def has_perm(self, perm, obj=None):
    return True

def has_module_perms(self, app_label):
    return True

objects = UserManager()

```

```

@classmethod
def get_user_obj(cls, username):
    return cls.objects.get(username=username)

@receiver(post_delete, sender=User)
def submission_delete(sender, instance, **kwargs):
    instance.profile_pic.delete(False)

class UserBlockList(models.Model):
    current_user = models.OneToOneField(User,
on_delete=models.CASCADE, related_name='user_block_list')
    blocked_user = models.ManyToManyField(User,
related_name='blocked_users')

    objects = models.Manager()

    def __str__(self):
        return str(self.current_user) + "blocked_users_obj"

POST_NOTIF_CHOICES = (('Disable', 'Disable'), ('From People I
Follow', 'From People I Follow'), ('From Everyone', 'From
Everyone'))

class Account_Settings(models.Model):
    user = models.OneToOneField(User,
on_delete=models.CASCADE, related_name="user_setting")
    disable_all = models.BooleanField(default=False)
    p_likes = models.CharField(verbose_name='Post Likes',
max_length=21, choices=POST_NOTIF_CHOICES,
default=POST_NOTIF_CHOICES[2][1])
    p_comments = models.CharField(verbose_name='Post
Comments', max_length=21, choices=POST_NOTIF_CHOICES,
default=POST_NOTIF_CHOICES[2][1])

```

```

    p_comment_likes = models.CharField(verbose_name='Post
Comment Likes', max_length=21,
choices=POST_NOTIF_CHOICES,
default=POST_NOTIF_CHOICES[0][1])

```

```

    f_requests = models.BooleanField(default=False)

```

```

    private_acc = models.BooleanField(default=False, null=True)

```

```

    activity_status = models.BooleanField(default=True, null=True)

```

```

objects = models.Manager()

```

```

class Meta:

```

```

    verbose_name = "User Setting"

```

```

def __str__(self):

```

```

    return str(self.user)

```

```

class Friends(models.Model):

```

```

    current_user = models.ForeignKey(User,
on_delete=models.CASCADE, related_name='current_user')

```

```

    followers = models.ManyToManyField(User)

```

```

    pending = models.ManyToManyField(User,
related_name='pending_requests')

```

```

    following = models.ManyToManyField(User,
related_name='following')

```

```

objects = models.Manager()

```

```

@classmethod

```

```

def follow(cls, current_user, follow_user):

```

```

    # current_user -> following +1

```

```

    friend = cls.objects.get(current_user=current_user)

```

```

    friend.following.add(follow_user)

```

```
# follow_user -> follow +1
friend= cls.objects.get(current_user=follow_user)
friend.followers.add(current_user)
```

```
@classmethod
```

```
def unfollow(cls, current_user, unfollow_user):
    # current_user -> following -1
    friend= cls.objects.get(current_user=current_user)
    friend.following.remove(unfollow_user)
```

```
#unfollow_user -> follow -1
friend = cls.objects.get(current_user=unfollow_user)
friend.followers.remove(current_user)
```

```
@classmethod
```

```
def add_to_pending(cls, current_user, pending_user):
    # add the current_user to pending_user's pending list.
    friend = cls.objects.get(current_user=pending_user)
    friend.pending.add(current_user)
```

```
@classmethod
```

```
def rm_from_pending(cls, current_user, pending_user):
    # remove the current_user from pending_user's pending list.
    # This function is called when the current_user wants to
    # cancel any friend request sent to pending_user.
    # (Private Account)
    friend = cls.objects.get(current_user=current_user)
    friend.pending.remove(pending_user)
```

```
@classmethod
```

```

def get_friends_list(cls, current_user):
    friend_obj, created =
cls.objects.get_or_create(current_user=current_user)
    online_friends, followers, following = None, None, None
    if not created:
        #online_friends = friend_obj.following.filter(active=True,
user_setting__activity_status=True).only('username', 'full_name',
'profile_pic')
        followers = friend_obj.followers.only('username',
'full_name', 'profile_pic')
        following =
friend_obj.following.filter(user_setting__activity_status=True).onl
y('username', 'profile_pic', 'full_name', 'last_login', 'active')
    return (followers, following)

class Meta:
    verbose_name = 'Relation'

```