SPAMMER DETECTION AND FAKE USER IDENTIFICATION ON SOCIAL NETWORKS

A PROJECT REPORT

Submitted by

S.SAI JAISHREE 211420104236

R.THANUSHA 211420104286

T.MUGILANJALI 211420104329

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2024

SPAMMER DETECTION AND FAKE USER IDENTIFICATION ON SOCIAL NETWORKS

A PROJECT REPORT

Submitted by

S.SAI JAISHREE 211420104236

R.THANUSHA 211420104286

T.MUGILANJALI 211420104329

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL 2024

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"SPAMMER DETECTION AND FAKE USER IDENTIFICATION ON SOCIAL NETWORKS"**is the bonafide work of **"S.SAI JAISHREE[211420104236], R.THANUSHA[211420104286] and T.MUGILANJALI [211420104329]"** who carried out the project work under my supervision.

Signature of the HOD with date

Dr.L.JABASHEELA M.E.,Ph.D.,
Professor & Head,

Department of Computer Science and Engineering,
Panimalar Engineering College,
Chennai-123.

Signature of the Supervisor with date

Mrs. A.KANCHANA M.E.,(Ph.D).,
Assistant professor,

Department of Computer Science and Engineering,
Panimalar Engineering College,
Chennai-123.

Submitted for the Project Viva-Vice examination held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION BY THE STUDENT

We **S.SAI JAISHREE [211420104236]**, **R.THANUSHA [211420104286]** and **T.MUGILANJALI [2114201042329]** here by declare that this project report titled "SPAMMER DETECTION AND FAKE USER IDENTIFICATION ON SOCIAL NETWORKS" under the guidance of **Mrs.A.KANCHANA M.E.,(Ph.D)** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

S.SAI JAISHREE

R.THANUSHA

T.MUGILANJALI

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**.,for his benevolent words and fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We want to express our deep gratitude to our Directors,**Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express my heartfelt thanks to **Dr. L. JABASHEELA,M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to Project Coordinator **Dr.K.VALARMATHI,M.E.,Ph.D** and Project Guide **Mrs.A.KANCHANA,M.E.,(Ph.D)** and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

S.SAI JAISHREE
R.THANUSHA
T.MUGILANJALI

ABSTRACT

Rumors spread dramatically fast through online social media services, and people are exploring methods to detect rumors automatically.Existing methods typically learn semantic representations of all reposts to a rumor candidate for prediction. However, it is crucial to efficiently detect rumors as early as possible before they cause severe social disruption, which has not been well addressed by previous works. In this paper, we present a novel early rumor detection model and automatically block the spammer or fake user, Credible Early Detection (CED). By regarding all reposts to a rumor candidate as a sequence, the proposed model will seek an early point-in-time for making a credible prediction. We conduct experiments on three real-world datasets, and the results demonstrate that our proposed model can remarkably reduce the time span for prediction by more than 85%, with better accuracy performance than all state-of-the-art baselines.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

S.NO.	ABBREVIATION	EXPANSION
1	CED	Credible Early Detection
2	CAMM	Cross-Attention Multi-Modal
3	SVM	Support Vector Machine
4	CNN	Convolutional Neural Network
5	RNN	Recurrent Neural Network
6	NLP	Natural Language Processing
7	HAM	Homotopy Analytical Method
8	JVM	Java Virtual Machine
9	GPL	General Public License

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

This is an overview of a research paper focused on detecting rumors in social media platforms, emphasizing the need for early detection to prevent significant social disruption. The paper introduces a new model called Credible Early Detection (CED) to address this challenge. It highlights the shortcomings of existing methods in early detection and presents CED as a solution by treating reposts to rumor candidates as a sequence and aiming to make credible predictions at an early point-in-time. The proposed model significantly reduces the time span for prediction while maintaining better accuracy compared to state-of-the-art baselines. Additionally, the passage categorizes existing rumor detection methods into three categories based on the types of information and methods used. It also underscores the significance of rumors as a social phenomenon, their rapid spread on social media platforms, and the challenges they pose due to their veracity being difficult to verify quickly. Rumor is an important phenomenon in social science and has witnessed many interests of researchers in social psychology field for many decades . According to the explanation of wikipedia1 and sociologists , a rumor usually involves some concerned public statements whose integrity cannot be quickly or ever verified. With the rapid growth of large-scale social media platforms, such as Facebook, Twitter, and Sina Weibo, rumor is becoming a more and more serious social problem than ever before. Due to the convenience of accessing information on these social media platforms, rumors can spread explosively within a short time before contradicted or detected.

1.2 PROBLEM DEFINITION

The problem outlined in the passage is the rapid spread of rumors through online social media platforms, which can lead to severe social disruption. While existing methods for detecting rumors typically focus on learning semantic representations of reposts to a rumor candidate, they often fail to efficiently detect rumors at an early stage. Detecting rumors early is crucial to mitigate their impact. To address this problem, the paper proposes a novel early rumor detection model called Credible Early Detection (CED). This model aims to automatically identify and block spammers or fake users by treating all reposts to a rumor candidate as a sequence and seeking an early point-in-time for making credible predictions.

CHAPTER 2
LITERATURE REVIEW

(1) TITLE AND JOURNAL:

CAMM:Cross-Attention Multi-modal Classification of Disaster-Related Tweets(2022)

AUTHORS:

ANURADHA KHATTAR AND S. M. K. QUADRI 1

DESCRIPTION:

The study integrates text and image data from social media posts during disaster to create CAMM(Cross-AttentionMulti-Modal), a multi-modal deep learning model using the attention mechanism. CAMM out performs uni-modal and popular multi modal models like MUTAN and BLOCK, achieving an average F1-score of 84.08%.

ADVANTAGES:

- The CAMM model introduces a novel approach to multi-modal classification, outperforming both uni-modal and existing multi-modal models.
- CAMM achieves a significantly higher F1-score compared to other models, demonstrating its effectiveness in disaster data classification.

DISADVANTAGES:

- During the past decade, social media platforms have been extensively used for information dissemination by the affected community and humanitarian agencies during a disaster.

(2) TITLE AND JOURNAL:

Enhancing Spam Comment Detection on Social Media With Emoji Feature and Post-Comment Pairs Approach Using Ensemble Methods of Machine Learning(2023).

AUTHORS:

ANTONIUS RACHMAT CHRISMANTO 1,2, ANNY KARTIKA SARI 1, AND YOHANES SUYANTO

DESCRIPTION:

The research addressed social media text spam detection using emojis and stacked post-comment pairs.Results showed improved accuracy and F1 scores, with SVM (RBF kernel) and soft voting ensemble methods performing best on average.

ADVANTAGES:

- Ensemble methods, such as soft voting, combine the predictions of multiple models, leading to more robust and accurate spam detection systems.

- Ensemble methods can leverage the strengths of different base models, potentially capturing a broader range of spam characteristics and improving overall performance.

DISADVANTAGES:

- Although some social media platforms already have spam filters, these are limited to English.
- Another problem is the limited publicly available datasets for identifying spam text on social media.

(3) TITLE AND JOURNAL:

Deep Feature Fusion for Rumor Detection on Twitter(2021)

AUTHORS:

ZHIRUI LUO , QINGQING LI , AND JUN ZHENG

DESCRIPTION:

The paper introduces a new method for Twitter rumor detection by combining linguistic features from source tweets with propagation tree patterns, using a pre-trained Transformer-based model and a CNN architecture.

ADVANTAGES:

- Introduces a novel method combining linguistic features of tweet text and propagation tree patterns for more effective rumor detection.
- Introduces a novel method to embed propagation tree patterns into a vector space, enabling extraction of temporal-structural features.

DISADVANTAGES:

- Most content-based approaches utilize some traditional linguistic features such as topic features, term frequency and bag-of-word for rumor detection.

(4) TITLE AND JOURNAL:

HANN: Hybrid Attention Neural Network for Detecting Covid-19 Related Rumors

AUTHORS:

ABDULQADER M. ALMARS 1, MALIK ALMALIKI1, TALAL H. NOOR 1, MAJED M. ALWATEER 1, AND ELSAYED ATLAM

DESCRIPTION:

The paper presents a Hybrid Attention Neural Network (HANN) for detecting rumors on social media, combining CNNs and Bi-LSTM networks with attention modules. Integrates software engineering features for enhanced accuracy**.**

ADVANTAGES:

- Introduces a hybrid attention neural network (HANN) that not only identifies rumors but also provides explanations for the model's decisions, enhancing interpretability.
- Enhances model accuracy by incorporating software engineering features such as followers, friends, and registration age, which further improve rumor detection effectiveness.

DISADVANTAGES:

- When analyzing text for text classification, not every word in a sentence has the same importance.
- These deep learning approaches can classify or detect information to rumors or non-rumors without explaining why the mode reached these decisions.

(5) TITLE AND JOURNAL:

Modified Genetic Algorithm for Feature Selection and Hyper Parameter Optimization: Case of XGBoost in Spam Prediction

AUTHORS:

NAZEEH GHATASHEH 1, ISMAIL ALTAHARWA 2, AND KHALED ALDEBEI

DESCRIPTION:

The paper proposes a modified genetic algorithm to address Twitter spam, combining dimensionality reduction and hyper parameter optimization. It utilizes extreme Gradient Boosting to create a highly accurate spam prediction model, outperforming traditional methods and even BERT-based deep learning models.

ADVANTAGES:

- Introduces a modified genetic algorithm for simultaneous dimensionality reduction and hyper parameter optimization, providing an efficient solution for model building.
- Utilizes less than 10% of the total feature space while maintaining high prediction performance, demonstrating efficient resource utilization.

DISADVANTAGES:

- Earlier solutions were limited to the rule-based and regular expression matching.

CHAPTER 3
THEORETICAL BACKGROUND

3.1 IMPLEMENTATION ENVIRONMENT

Compared with feature engineering methods, the effectiveness of deep neural networks on automatic feature learning has been verified in many NLP tasks, such as parsing, text classification, machine translation and question answering. Employed Recurrent Neural Network (RNN) to learn a dynamic temporal representation for each microblog based on its reposts over time and make the prediction according to the representation of the entire repost sequence. It is the first attempt to introduce deep neural networks into repost-based rumor detection and achieves considerable performance on real-world datasets. Employed paragraph vector and convolutional neural networks on the repost sequence to detect rumors.Furthermore, existing methods may not adequately prioritize the early detection of rumors, which is crucial for mitigating their impact. As a result, there is a need for a more sophisticated approach that can pinpoint rumors at an early stage and automatically take action to prevent their dissemination. Overall, the current systems for rumor detection in online social media platforms face challenges in efficiently detecting and addressing rumors before they cause widespread social disruption. The proposed Credible Early Detection (CED) model aims to address these shortcomings by treating reposts to a rumor candidate as a sequence and seeking an early point-in-time for making credible predictions.

3.2 SYSTEM ARCHITECTURE

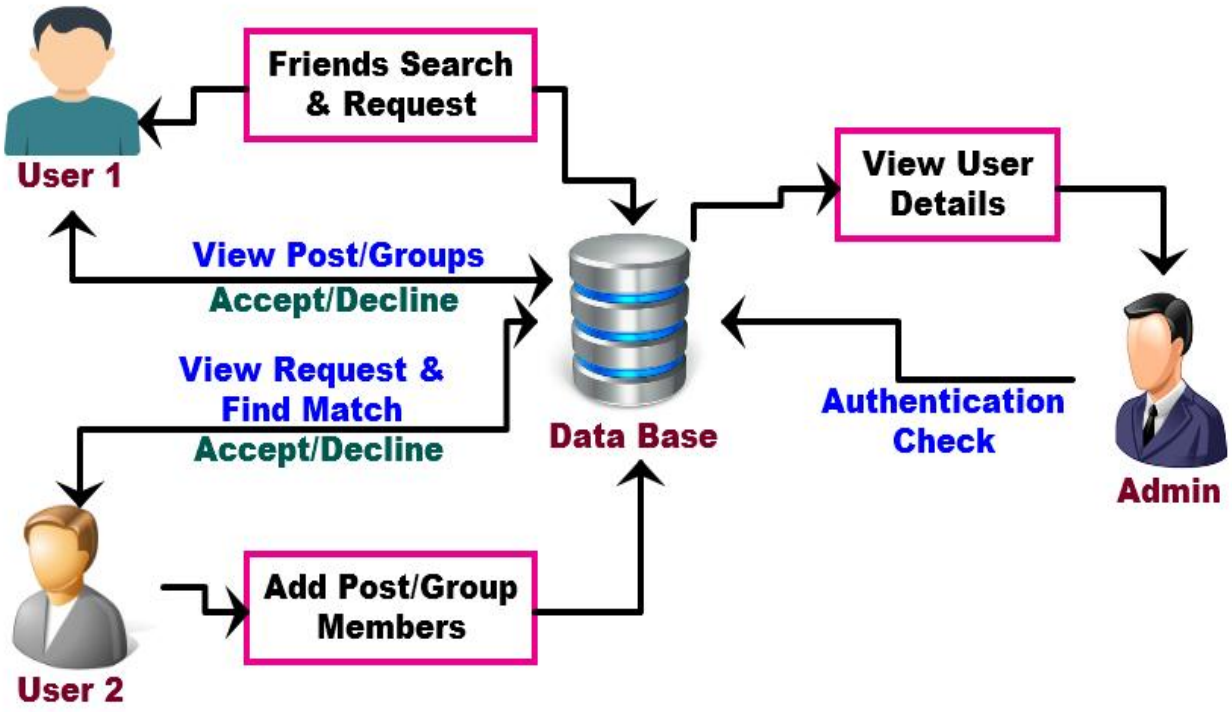


Fig.1 SYSTEM ARCHITECTURE

This system is designed to automatically detect rumors spreading on social media platforms. The system, named Credible Early Detection (CED), aims to identify rumors early and block the accounts spreading them. Social Media Service represents the platform where rumors spread, like Twitter or Facebook. Users interact with this service by generating content (text, images, videos) and engaging with existing content. Rumor Candidate Identification likely identifies potential rumors based on predefined criteria. It could analyze content characteristics, user behavior, or engagement patterns. Repost Stream Generation block gathers and sequences reposts associated with the identified rumor candidate. This likely involves collecting content reposted from the original source and analyzing the sequence of reposts over time. Credible Early Detection Model is the core of the system. If the CED model classifies the content as a rumor, Rumor Spreader Block takes action to suppress its spread. This could involve blocking the spreader's account, hiding the content, or issuing warnings to users.

3.3 PROPOSED METHODOLOGY

In this article, we propose a novel model to implement online real-time detection of social media rumors.we present the temporal repost sequence of a specific rumor and the corresponding prediction probability curve. For this example, we can make a credible prediction at an early time stamp, as there appears quite a lot doubts and refutation to the original microblog. Based on this observation, we introduce “Credible Detection Point” and propose a novel early rumor detection model, Credible Early Detection (CED). Specifically, CED learns to determine the “Credible Detection Point” for each repost sequence during the training stage and ensure the credibility of the prediction result at this time point, i.e., there is no plot reversal after credible detection point.Unlike existing methods that primarily focus on learning semantic representations of reposts related to a rumor candidate, CED takes a novel approach by considering all reposts as a sequence.CED prioritizes the early detection of rumors by seeking an early point-in-time to make credible predictions. This is crucial for mitigating the impact of rumors before they spread widely. In addition to rumor detection, CED automatically identifies and blocks spammers or fake users contributing to the dissemination of rumors. This helps in controlling the spread of misinformation at its source.By treating all reposts to a rumor candidate as a sequence, CED leverages sequential information to make predictions.

3.4 ER DIAGRAM

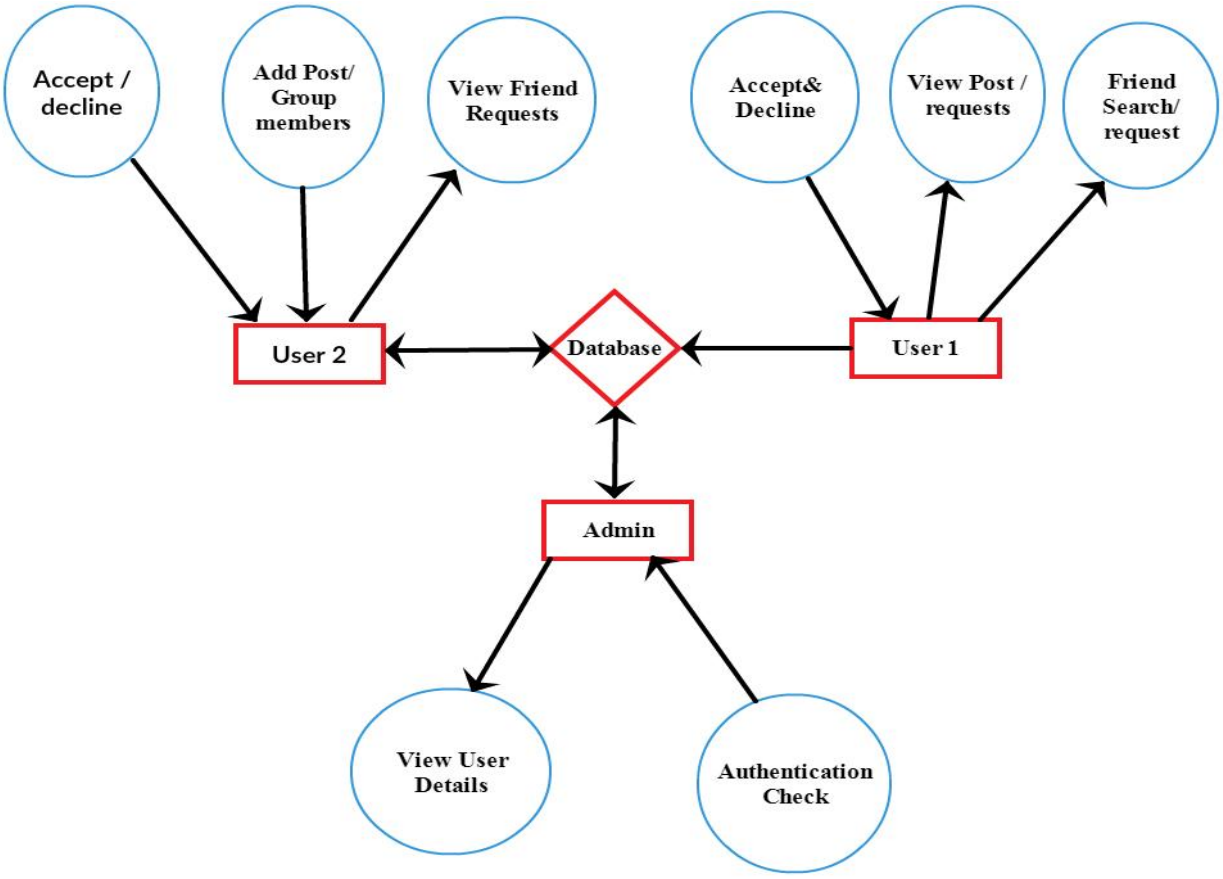


Fig.2 ER DIAGRAM

An ER diagram, which stands for **Entity-Relationship Diagram**, is a graphical representation that shows the relationships between different entities (things, concepts, or objects) within a system, typically a database. It uses specific symbols like rectangles, diamonds, and lines to depict these entities, their attributes (properties), and how they connect with each other. The process starts with User 1, who can either accept or decline a friend request from User 2. If User 1 accepts the request, they become friends and can see each other's posts. If User 1 declines the request, they are not friends and cannot see each other's posts.If User 1 wants to add User 2 as a friend, they can search for them by

name or email address. If they find User 2, they can send them a friend request. User 2 can then accept or decline the request.The diagram also shows that users can view a list of their friends and manage their friend requests. Overall, the diagram is a helpful guide to understanding how to add friends on social media.

3.5 DATA FLOW DIAGRAM

DFD 0

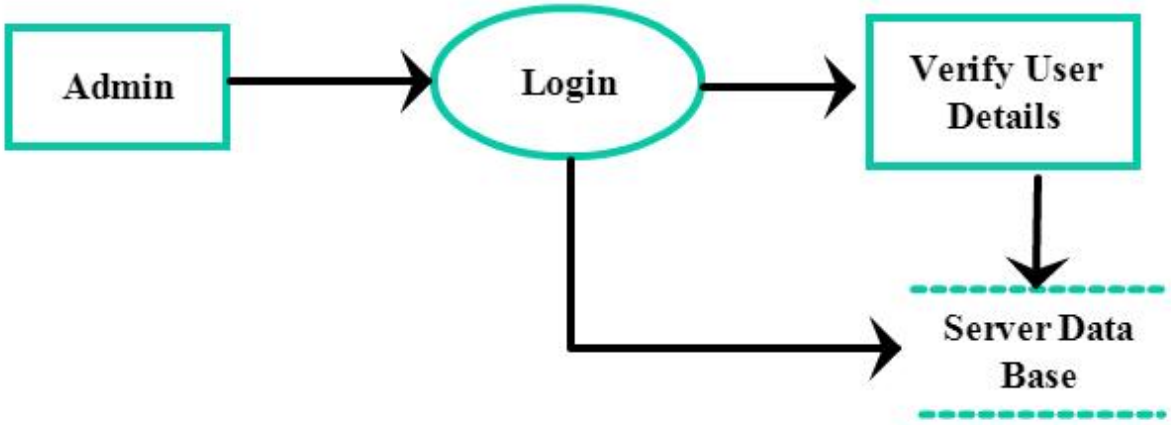


Fig.3 DFD 0

DFD 1

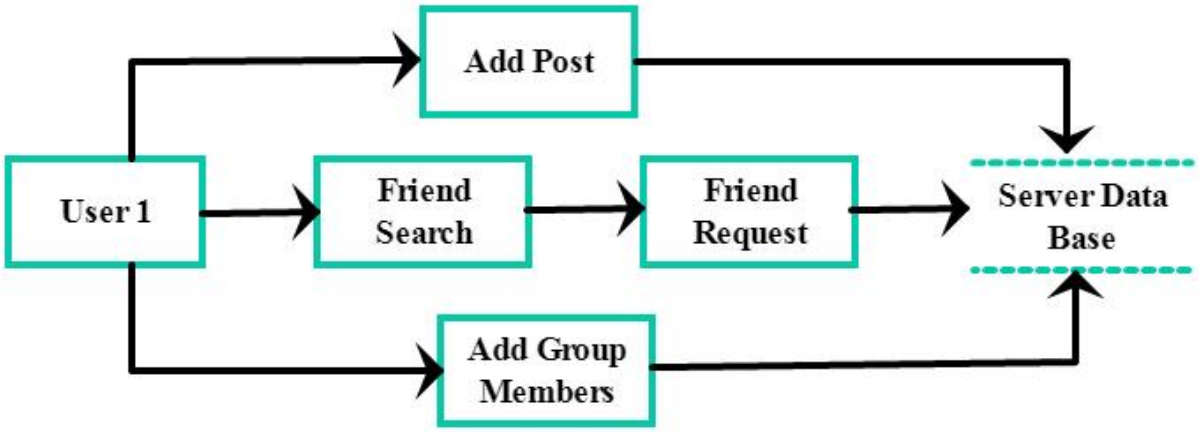


Fig.4 DFD 1

DFD 2

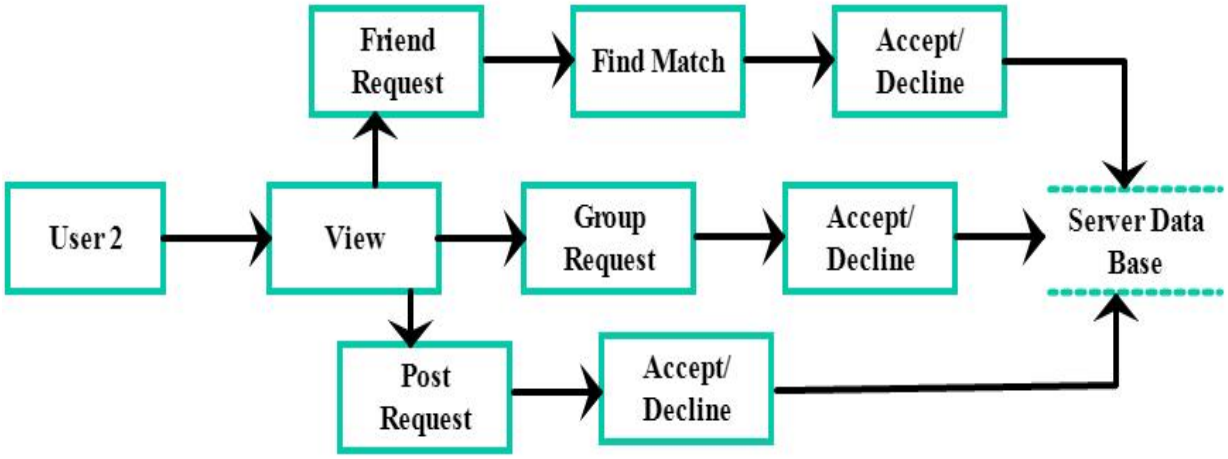


Fig.5 DFD 2

OVERALL DFD

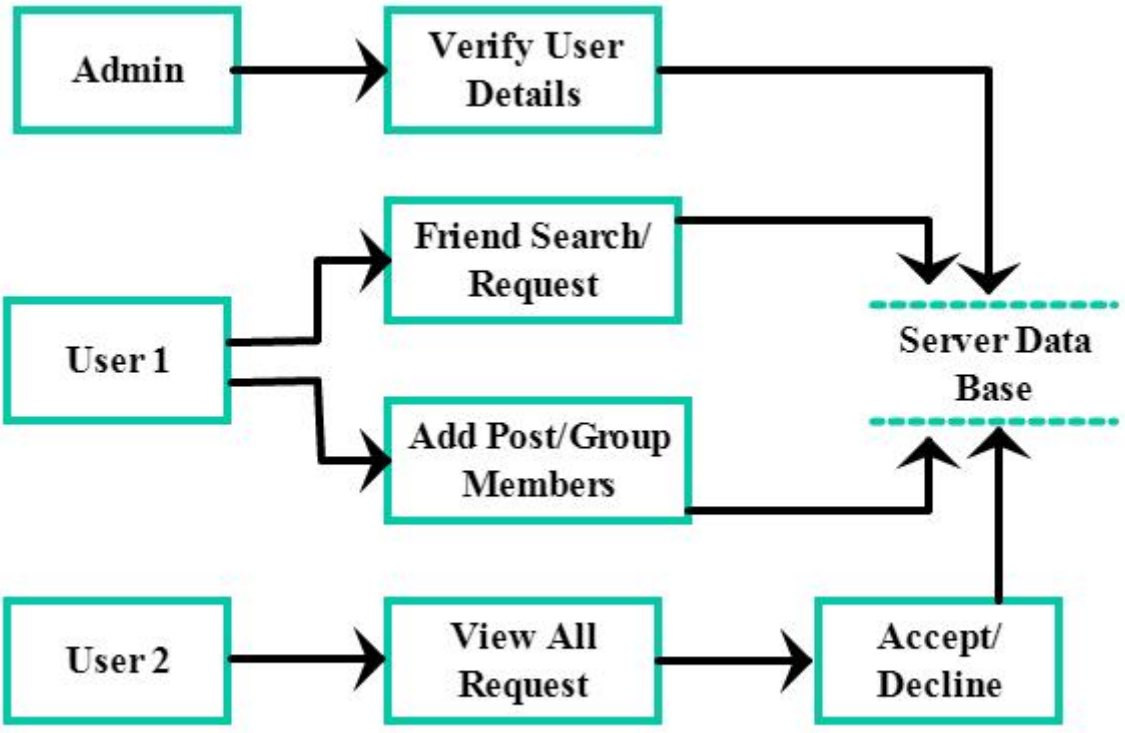


Fig.6 OVERALL DFD

3.6 UML DIAGRAMS

3.6.1 USECASE DIAGRAM

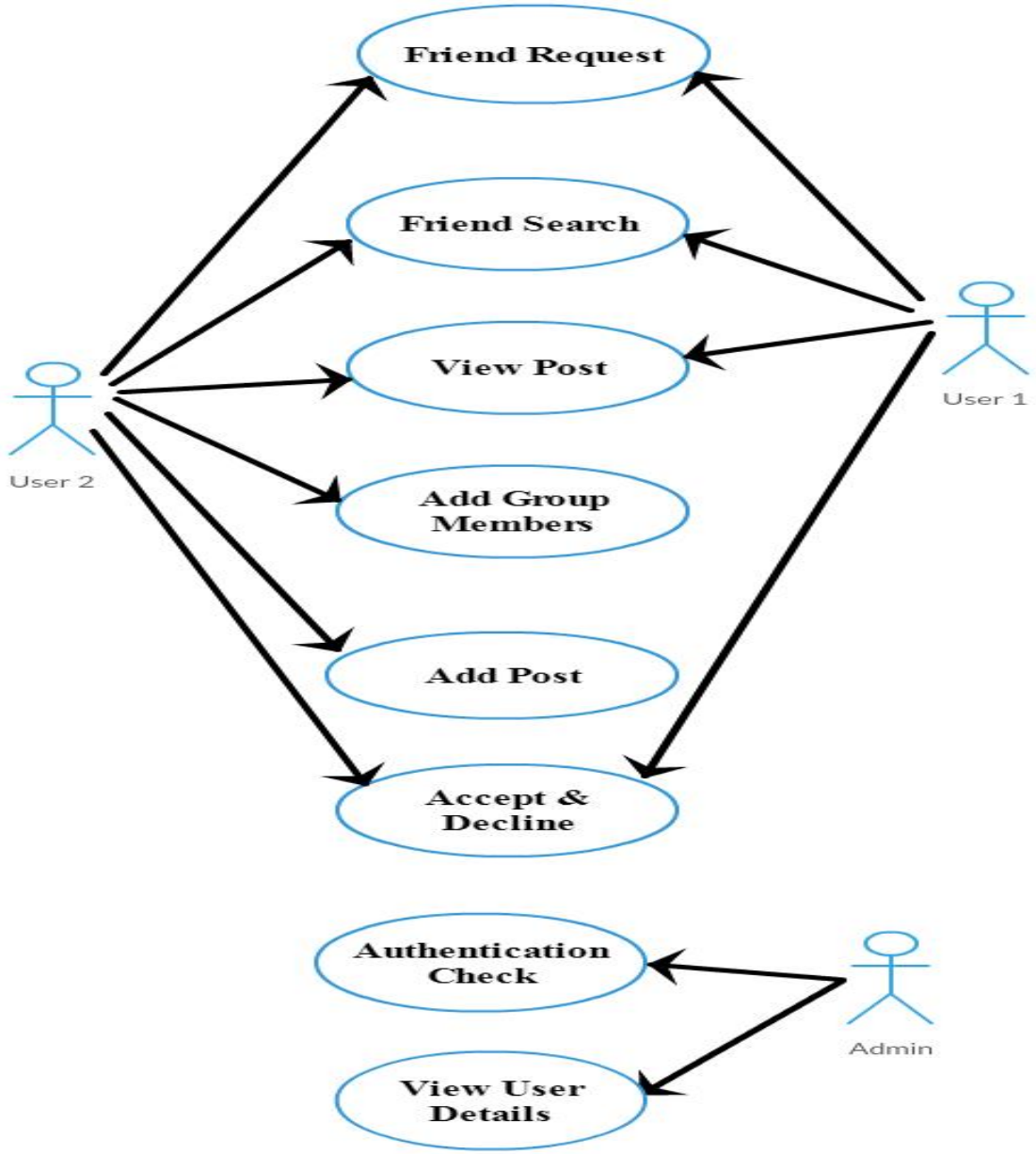


Fig.7 USECASE DIAGRAM

A **use case diagram** is a visual representation of the interactions between a system and its users or external systems. It's a fundamental tool in the Unified Modeling Language (UML) for capturing system requirements and functionality.The process starts with a user (User 1) who can either accept or decline a friend request sent by another user (User 2).If User 1 accepts the request, they become friends and can see each other's posts. If User 1 declines the request, they don't become friends and cannot see each other's posts. If User 1 wants to add User 2 as a friend, they can search for them by name or email address using the "Friend Search" option. Once they find User 2, they can send them a friend request. User 2 can then accept or decline the request, as mentioned before. The diagram also shows that users can view a list of their friends and manage their friend requests. They can do this by going to their "Friends" section or clicking on "View User Details" and then "Friends." Users can also view the profiles of other users, even if they are not friends, by clicking on "View User Details." However, they may not be able to see all of the other user's posts or information depending on their privacy settings. There is a step labelled "Authentication Check" near the beginning of the process. This likely refers to some security measures in place to verify the identity of users and prevent fraudulent activity. The diagram provides a general overview of how to add friends and manage friend requests on a social media platform. It's important to note that the specific details of the process may vary depending on the platform you're using.

3.6.2 CLASS DIAGRAM

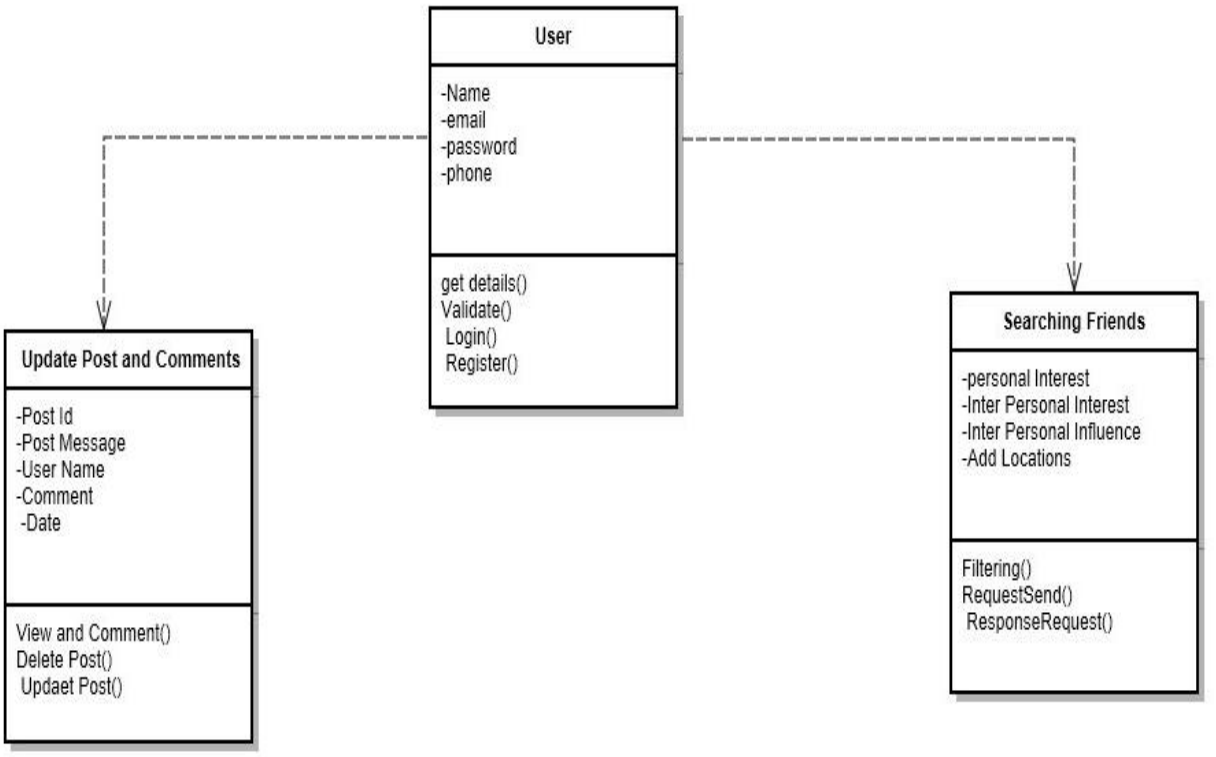


Fig.8 CLASS DIAGRAM

A class diagram is a type of structural diagram used in the Unified Modeling Language (UML) to visually represent the classes, attributes, methods, and relationships within a software system. It acts as a blueprint, capturing the static structure of the system and showing how its various components interact. This diagram represents the platform where rumors spread, such as Twitter or Facebook. Users interact with this service by generating content (text, images, videos) and engaging with existing content. This block likely identifies potential rumors based on predefined criteria. It could analyze content characteristics, user behavior, or engagement patterns. This block gathers and sequences reposts associated with the identified rumor candidate. This likely involves collecting content reposted from the original source and analyzing the sequence of reposts over time. This is the core of the system. It analyzes the repost stream using machine learning techniques to

predict if the content is a rumor with high confidence at an early stage. If the CED model classifies the content as a rumor, this block takes action to suppress its spread. This could involve blocking the spreader's account, hiding the content, or issuing warnings to users.

3.6.3 ACTIVITY DIAGRAM

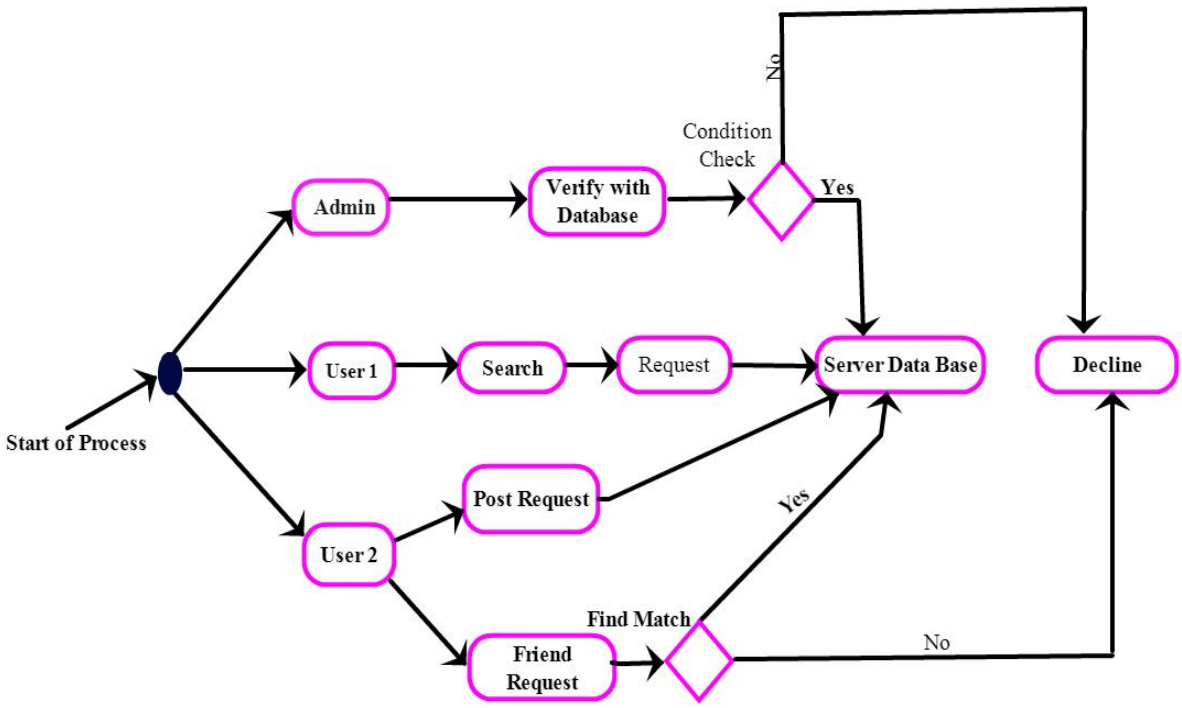


Fig.9 ACTIVITY DIAGRAM

This diagram represents the initial process of creating a new customer account. It likely involves gathering customer information, verifying their identity, and setting up their accounts for various banking services. This block covers activities related to managing existing accounts, such as updating personal information, changing passwords, or requesting new services. This block handles all financial transactions initiated by customers, including deposits, withdrawals, payments, and money transfers. This block monitors account activity for suspicious patterns that might

indicate fraud attempts. It may also involve implementing security measures to prevent unauthorized access and protect customer information. This block provides assistance to customers with any questions or issues they encounter while using their bank accounts. This could involve responding to inquiries through various channels like phone, email, or chat. This block ensures the bank adheres to all relevant financial regulations and reporting requirements.

3.6.4 SEQUENCE DIAGRAM

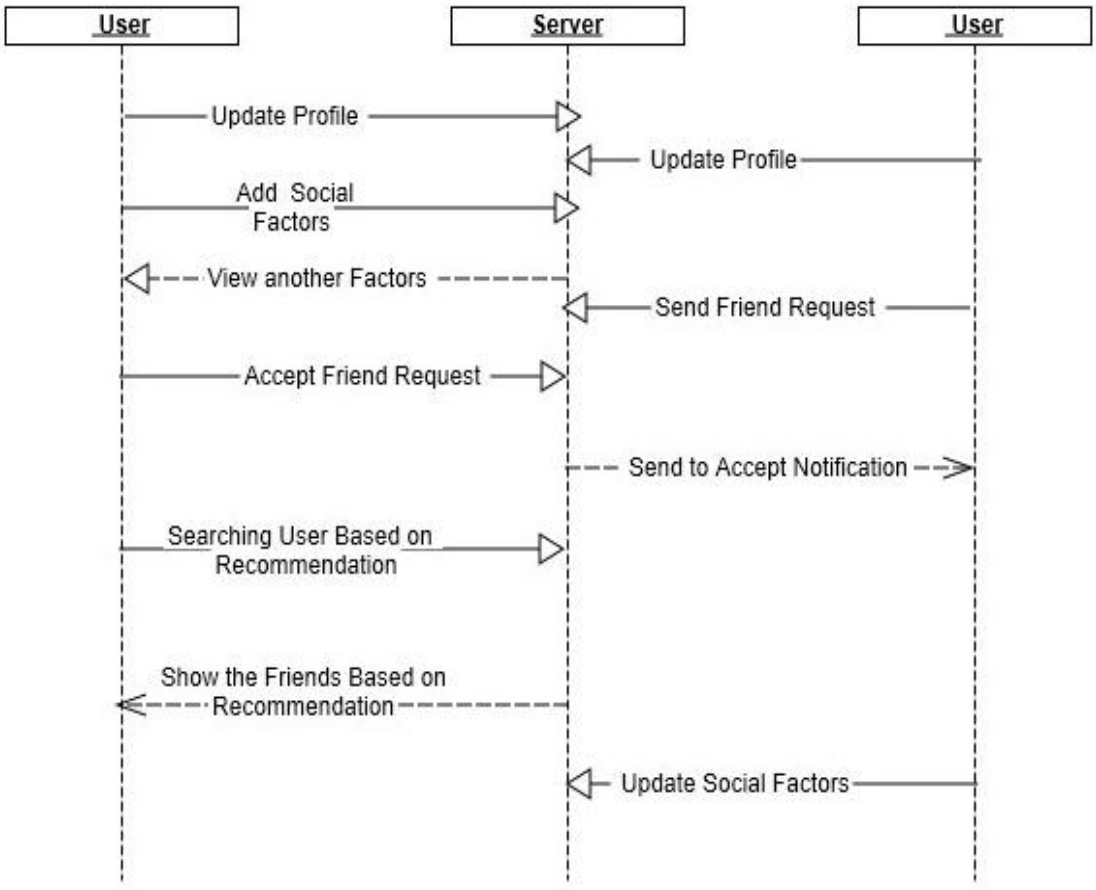


Fig.10 SEQUENCE DIAGRAM

A sequence diagram, also known as an event diagram or interaction diagram, is a type of Unified Modeling Language (UML) diagram specifically designed to visualize the interactions between objects in a system over time. It shows the

sequential order of messages exchanged between objects participating in a specific interaction or scenario. User 1 **r**epresents the user initiating the friend request. User 2 represents the user receiving the friend request. Server represents the social media platform's server that processes the request. User 1 sends a "Send Friend Request" message to the server, including User 2's profile information. The server sends a "Receive Friend Request" message to User 2, containing User 1's profile information and the option to accept or decline the request. User 2 responds with either an "Accept Friend Request" or "Decline Friend Request" message to the server. The server sends a "Friend Request Accepted" or "Friend Request Declined" message to User 1, depending on User 2's response. The server updates User 2's profile to reflect the new friend connection (if the request was accepted) or removes the pending request (if declined). The server might send additional notifications to both users, such as "You are now friends with [User Name]" or "Your friend request to [User Name] has been declined.

3.7 HARDWARE ENVIRONMENT

- Processor – i3,i5,i7
- RAM – 4GB
- Hard Disk – 260GB

3.8 SOFTWARE ENVIRONMENT

- Operation System – 7/8/10
- Front-end - Java
- Back-end - MySql
- Tool - Netbeans 7.3.1

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1 MODULE DESIGN SPECIFICATION

There are seven modules:

- Registration & Login
- Timeline Add
- Friend Request
- Authentication
- Events in circle
- Post visuals
- Chat

4.1.1 REGISTRATION & LOGIN

In this module normal users who want to like together with peoples in this site then create an account on this site by executing registration process,means normal users are provide basic details like user name,password,address,e-mail id and also phone number. After registration if the user want to access account then enter correct user name / e-mail id and password. If credentials are correct then then server allows to go to in side the websites or else user name or password alert is generated by server.

4.1.2 TIMELINE ADD

In this modules User post some image contents for share him / her feelings to other peoples means share within friends lists.This post will be displayed on the timeline of him / her friends list.

4.1.3 FRIEND REQUEST

In this modules User enter some of the string into the search bar and then sent this string as request to the server.When receive this type of requests then server automatically check the possibility of results and then response to the requested user.This response has only name of the persons, does not contain another information.If user want to friend any member from this lists then select parameters and then send friend request.

4.1.4 AUTHENTICATION

In this module ,users are all will register in this websites with own details like user name, password, Addressee-Mail ID and also with user type like normal users. After registration ito access their panel then enter correct user name / e-mail id and password it will allows to go to inside the websites or else user name or password alert will come. Admin will enter correct user name and password it will allows to go to Admin home the websites. If user name and password is incorrect then it will not allow to access their panel.

4.1.5 EVENTS IN CIRCLE

In this module each and every users has a permission to add events of particular location of network with the following details event name, exact place and time. Those details are stored into database. This event are also notified to every users with in that specified location.

4.1.6 POST VISUALS

In this module user of social network will exchange the visual like photos, videos and other things. Here users also provide like and commands for those visuals as the feedback

4.1.7 CHAT

In this Module users have a n number of friends by send and accept friend request to another users. The users want to be a friend then accept friend requests. After accept friend requests want chat / communicate with those who are all friends. Then select users from lists and then chat with that selected users.

4.2 ALGORITHM

4.2.1 Ham Algorithm Description

Content-Based Filtering Technique Algorithms analyze words, the occurrence of words, and the distribution of words and phrases inside the content of e-mails and segregate them into spam non-spam categories .Case Base Spam Filtering Method Algorithms trained on well-annotated ham/non-spam marked emails try to classify the incoming mails into two categories. Heuristic or Rule-Based ham Filtering Technique

Algorithms use pre-defined rules in the form of a regular expression to give a score to the messages present in the e-mails. Based on the scores generated, they segregate emails into ham non-spam categories. The Previous Likeness Based Spam filtering Technique Algorithms extract the incoming mails' features and create a multidimensional space vector and draw points for every new instance. Based on the naïve bayes classifier algorithm, these new points get assigned to the closest class of ham and non-spam. Adaptive Spam Filtering Technique Algorithms classify the incoming mails in various groups and, based on the comparison scores of every group with the defined set of groups, ham and non-spam emails got segregated.

STEPS:

Step 1: E Mail Data Collection.

Step 2: Preprocessing Email Content.

Step 3: Feature Extraction And Selection.

Step 4: Ham Implementation.

Step 5: Performance Analysis.

ALGORITHM :

Procedure HAM

d-dini; nlead=0;iter-0 initial solution: PWE2 Cmax=Cmax; x = x;

repeat

nml1-0

if random<ß then

indmet=0

else indmet = 1

endif

do

nml1=nml1+1;

if indmet=0 then

LSI

else

LS2

endif

if Cmax<Cmaxo or nml1=1 then indmet= 1 - indmet.

else exit do

endif

loop

<Cmax then Cox - Cux; x = xendif if Cmax <Cm and random <a then

if Ca max

endif

x= x;

x"-deconstruct(x)

x-construct(x") if Cmax<Cmx then

C-Cm x-x endif

end

until stopping criterion is met

4.2.2 MATCHING ALGORITHM

Aho-Corasick algorithm:

Aho–Corasick algorithm is a string-searching algorithm. It is a kind of dictionary- matching algorithm that locates elements of a finite set of strings within an input text. It matches all strings simultaneously. The complexity of the algorithm is linear in the length of the strings plus the length of the searched text plus the number of output matches. Note that because all matches are found, there can be aquadratic number of matches if every sub-string matches.

CHAPTER 5
RESULTS & DISCUSSION

5.1 PERFORMANCE PARAMETERS/TESTING

5.1.1 TESTING OBJECTIVES

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding error, if it exists. The tests are inadequate to detect possibly present errors. The software more or less confirms to the quality and reliable standards

5.1.2 TESTING LEVELS

System testing is stage of implementation which is aimed at ensuring that the system works accurately and efficient before live operation commences. Testing is vital the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

Unit Testing

In the lines of strategy, all the individual functions and modules were put to the test independently. By following this strategy all the errors in coding were identified and corrected. This method was applied in combination with the White and Black Box testing Techniques to find the errors in each module.

Integration Testing

Data can be lost across the interface; one module can have an adverse effect on others. Integration testing is a systematic testing for constructing program structure.

While at the same time conducting tests to uncover errors associated within the interface. Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order sets and conducted. The objective is to take unit tested modules and combine them test it as a whole. Thus, in the integration-testing step all the errors uncovered are corrected for the next testing steps.

Validation Testing

The outputs that come out of the system are as a result of the inputs that go into the system. The correct and the expected outputs that go into the system should be correct and proper. So this testing is done to check if the inputs are correct and they are validated before it goes into the system for processing.

Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for the user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes whenever required. This is done in regard to the following point:

- Input screen design
- Output screen design

An acceptance test has the objective of selling the user on the validity and reliability of the system. It verifies that the system's procedures operate to system specifications and that the integrity of important data is maintained. Performance of an acceptance test is actually the user's show. User motivation is very important for the successful performance of the system. After that a comprehensive report is prepared.

CHAPTER 6

CONCLUSION & FUTURE ENHANCEMENT

CONCLUSION

In this paper, we focus on the task of early detection of social media rumors. This task aims to distinguish a rumor as early as possible based on its repost information. While existing works can only make a prediction with the entire or fixed proportions of repost sequence, we assume there exists a “Credible Detection Point” for each microblog. Moreover, we propose Credible Early Detection (CED) model to remove the effect of interference repost information and make a credible prediction at a specific credible detection point. Experimental results on real-world datasets demonstrate that our model can significantly reduce the time span for predictions by more than 85%, with even better accuracy. For future works, we can incorporate other important information into early rumor detection, such as publisher’s profiles and propagation structure besides the repost information and original microblogs.

FUTURE ENHANCEMENTS

When the features (topological features only) were taken into consideration, we succeeded in achieving interesting results. The results suggest that the proposed approach can be used to forecast critical events that evolving communities may undergo. While the model as presented can achieve interesting results, several points still need further work in order to improve the approach. For instance, a limitation of our approach is that we treat critical events with equal importance. However, in some applications, different communities may have their own life cycles. Critical events should thus be treated accordingly. One possible direction is to adjust the calculation of the fluctuation by incorporating a weighting scheme in which the importance of events such as “appear” and “disappear” is not considered equal but based on the dynamics of communities.

APPENDICES

6.1 SOURCE CODE

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package BUSSINESS;

import DATA.DB;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.text.SimpleDateFormat;
import java.util.Iterator;
import java.util.List;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.http.HttpSession;
import org.apache.tomcat.util.http.fileupload.FileItem;
import org.apache.tomcat.util.http.fileupload.FileUploadException;
import org.apache.tomcat.util.http.fileupload.disk.DiskFileItemFactory;
import org.apache.tomcat.util.http.fileupload.servlet.ServletFileUpload;

/**
 *
 * @author alaguraj
 */
public class Register extends HttpServlet {

    /**
     * Processes requests for both HTTP
     * <code>GET</code> and
     * <code>POST</code> methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession(true);
        try {
            SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd h:mm:ss");
```

```
        java.util.Date de = new java.util.Date();
        String sdte = sdf.format(de);
        int price = 0;
        String usrcat = "", fname = "", lname = "", mail = "", gdnr = "", dob = "",
passwd = "", saveFile = "", fileName = "", cntry = "", state = "", city = "", addrss = "",
latilang = "";
        String contentType = request.getContentType();
        DiskFileItemFactory factory = new DiskFileItemFactory();
        factory.setSizeThreshold(4012);
        ServletFileUpload upload = new ServletFileUpload(factory);

        List item = null;
        try {
           item = upload.parseRequest(request);
        } catch (FileUploadException e) {
           e.printStackTrace();
        }
        byte[] data = null;
        Iterator iter = item.iterator();
        while (iter.hasNext()) {
           FileItem itemm = (FileItem) iter.next();
           if (itemm.isFormField()) {
              String name = itemm.getFieldName();
              String value = itemm.getString();
              if (name.equalsIgnoreCase("UType")) {
                 usrcat = value;
                 System.out.println("usrcat is  : " + usrcat);
              } else if (name.equalsIgnoreCase("fname")) {
                 fname = value;
                 System.out.println("fname is : " + fname);
```

```
} else if (name.equalsIgnoreCase("lname")) {
    lname = value;
    System.out.println("lname is : " + lname);
} else if (name.equalsIgnoreCase("Gndr")) {
    gdnr = value;
    System.out.println("gdnr is : " + gdnr);
} else if (name.equalsIgnoreCase("emailid")) {
    mail = value;
    System.out.println("mail is : " + mail);
} else if (name.equalsIgnoreCase("dteobth")) {
    dob = value;
    System.out.println("dob is  : " + dob);
} else if (name.equalsIgnoreCase("password")) {
    passwd = value;
    System.out.println("passwd is : " + passwd);
} else if (name.equalsIgnoreCase("ppic")) {
    data = itemm.get();
    fileName = itemm.getName();
} else if (name.equalsIgnoreCase("country")) {
    cntry = value;
    System.out.println("cntry is  : " + cntry);
} else if (name.equalsIgnoreCase("state")) {
    state = value;
    System.out.println("state is  : " + state);
} else if (name.equalsIgnoreCase("city")) {
    city = value;
    System.out.println("city is : " + city);
} else if (name.equalsIgnoreCase("adds")) {
    addrss = value;
    System.out.println("addrss is : " + addrss);
```

```
            } else if (name.equalsIgnoreCase("latillang")) {
                latilang = value;
                System.out.println("latilang is : " + latilang);
            } else {
                System.out.println(name);
            }
        } else {
            data = itemm.get();
            fileName = itemm.getName();
            System.out.println("File name is .....\t" + fileName);
        }
    }
    DB fb = new DB();
    ResultSet rs = fb.Select("select * from users where Mail_ID='" + mail + "'");
    if (rs.next()) {
        session.setAttribute("fbmsg", "Mail ID Alreday Presented Please Try Again
With Another Mail ID");
        response.sendRedirect("Register.jsp");
    } else {
        System.out.println(saveFile);
        saveFile = fileName;
        String path = request.getSession().getServletContext().getRealPath("/");
        String patt = path.replace("\\build", "");
        String np = patt + "\\AllUsers\\";
        String strPath = np + saveFile;
        File ff = new File(strPath);
        System.out.println("Fikle PATH\t" + strPath);
        FileOutputStream fileOut = new FileOutputStream(ff);
        fileOut.write(data, 0, data.length);
        fileOut.flush();
```

```
fileOut.close();
System.out.println(saveFile);
File image = new File(strPath);
FileInputStream fis = new FileInputStream(image);
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/social", "root", "admin");
PreparedStatement pstmnt = con.prepareStatement("insert into users(TYPE, F_Name, L_Name, Mail_ID, DOB, P_PIcture, PF_Name, Country, State, City, Address, Lat_Longtude, R_Time,Password,Gender) values(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");
pstmnt.setString(1, usrcat);
pstmnt.setString(2, fname);
pstmnt.setString(3, lname);
pstmnt.setString(4, mail);
pstmnt.setString(5, dob);
pstmnt.setBinaryStream(6, (InputStream) fis, (int) (image.length()));
pstmnt.setString(7, fileName);
pstmnt.setString(8, cntry);
pstmnt.setString(9, state);
pstmnt.setString(10, city);
pstmnt.setString(11, addrss);
pstmnt.setString(12, latilang);
pstmnt.setString(13, sdte);
pstmnt.setString(14, passwd);
pstmnt.setString(15, gdnr);
int tr = pstmnt.executeUpdate();
if (tr > 0) {
   session.setAttribute("fbmsg", "Registered SuccessFully");
   response.sendRedirect("index.jsp");
```

```
            } else {
                session.setAttribute("message", "ERROR TRY AGAIN");
                response.sendRedirect("Register.jsp");
            }
        }
    } catch (Exception e) {
        System.out.println(e);
        e.printStackTrace();
    } finally {
        out.close();
    }
}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
/**
 * Handles the HTTP
 * <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}
```

```
    /**
     * Handles the HTTP
     * <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

UserLog

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
```

```
 */
package BUSSINESS;

import DATA.DB;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class UsrLogCheck extends HttpServlet {

    /**
     * Processes requests for both HTTP
     * <code>GET</code> and
     * <code>POST</code> methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
```

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException, SQLException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    HttpSession session = request.getSession(true);
    SimpleDateFormat yt = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
    SimpleDateFormat yt1 = new SimpleDateFormat("yyyy-MM-dd");
    java.util.Date dt = new java.util.Date();
    String ste = yt.format(dt);
    String sdte1 = yt1.format(dt);
    String msg = "";
    DB gf = new DB();
    DB gf1 = new DB();
    try {
        String uname = request.getParameter("username");
        String upass = request.getParameter("password");
        if ((uname.equalsIgnoreCase("fbadmin@gmail.com"))||(uname.contains("admin"))) {
            ResultSet rs1 = gf1.Select("select * from admindet where Mail_ID='" + uname + "' and Password='" + upass + "'");
            if (rs1.next()) {
                if ((uname.equals(rs1.getString("Mail_ID"))) && (upass.equals(rs1.getString("Password")))) {
                    int lgin = gf.Insert("insert into logdet(L_UName,L_UINDate,L_UINTime) value('" + uname + "','" + sdte1 + "','" + ste + "')");
                    System.out.println("Normal User");
                    session.setAttribute("ADINTIME", ste);
                    session.setAttribute("ADUNAME", uname);
```

```
                session.setAttribute("ADUPASS", upass);
                session.setAttribute("fbmsg", "You are Loggedin Successfully...");
                response.sendRedirect("ADMNHome.jsp");
            } else {
                session.setAttribute("fbmsg", "Wrong Format Of Username And Password!!!...");
                response.sendRedirect("UsrLogin.jsp");
            }
        } else {
            session.setAttribute("fbmsg", "Invalid Username and Password!!!...");
            response.sendRedirect("UsrLogin.jsp");
        }

    } else {
        ResultSet rs = gf.Select("select * from users where Mail_ID='" + uname + "' and Password='" + upass + "'");
        System.out.println("select * from users where Mail_ID='" + uname + "' and Password='" + upass + "'");
        if (rs.next()) {
            if ((uname.equals(rs.getString("Mail_ID"))) && (upass.equals(rs.getString("Password")))) {
                if ((rs.getString("TYPE")).equals("Normal_User")) {
                    int lgin = gf.Insert("insert into logdet(L_UName,L_UINDate,L_UINTime) value('" + uname + "','" + sdte1 + "','" + ste + "')");
                    System.out.println("Normal User");
                    session.setAttribute("UINTIME", ste);
                    session.setAttribute("ULOCATION", rs.getString("City"));
                    session.setAttribute("UNAME", uname);
                    session.setAttribute("UPASS", upass);
```

```
                    session.setAttribute("fbmsg", "You are Loggedin Successfully...");
                    response.sendRedirect("UsrHome.jsp");
                } else if ((rs.getString("TYPE")).equals("Product_Distributor")) {
                    System.out.println("Product User");
                    int lgin = gf.Insert("insert into logdet(L_UName,L_UINDate,L_UINTime) value('" + uname + "','" + sdte1 + "','" + ste + "')");
                    session.setAttribute("PINTIME", ste);
                    session.setAttribute("PLOCATION", rs.getString("City"));
                    session.setAttribute("PNAME", uname);
                    session.setAttribute("PPASS", upass);
                    session.setAttribute("fbmsg", "You are Loggedin Successfully...");
                    response.sendRedirect("POnrHome.jsp");
                }
            } else {
                session.setAttribute("fbmsg", "Invalid Username and Password!!!...");
                response.sendRedirect("UsrLogin.jsp");
            }
        } else {
            session.setAttribute("fbmsg", "Invalid Username and Password!!!...");
            response.sendRedirect("UsrLogin.jsp");
        }
    } finally {
        out.close();
    }
}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
```

```
    /**
     * Handles the HTTP
     * <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            Logger.getLogger(UsrLogCheck.class.getName()).log(Level.SEVERE,     null,
ex);
        }
    }

    /**
     * Handles the HTTP
     * <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            Logger.getLogger(UsrLogCheck.class.getName()).log(Level.SEVERE,     null,
ex);
        }
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

Poupdate:

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package BUSSINESS;

import DATA.DB;
```

```
import DATA.GetInfo.MnubarDet;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.text.SimpleDateFormat;
import java.util.Iterator;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.apache.tomcat.util.http.fileupload.FileItem;
import org.apache.tomcat.util.http.fileupload.FileUploadException;
import org.apache.tomcat.util.http.fileupload.disk.DiskFileItemFactory;
import org.apache.tomcat.util.http.fileupload.servlet.ServletFileUpload;

/**
 *
 * @author alaguraj
```

```
 */
public class POUpdateProfile extends HttpServlet {

    /**
     * Processes requests for both HTTP
     * <code>GET</code> and
     * <code>POST</code> methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException, SQLException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession(true);
        SimpleDateFormat yt = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
        java.util.Date dt = new java.util.Date();
        String ste = yt.format(dt);
        String msg = "";
        DB gf = new DB();
        try {
            String umail = "", fname = "", lname = "", DOB = "", Adds = "", city = "", sname = "", splace = "", fileName = "", saveFile = "";
            String contentType = request.getContentType();
            DiskFileItemFactory factory = new DiskFileItemFactory();
            factory.setSizeThreshold(4012);
```

```
ServletFileUpload upload = new ServletFileUpload(factory);
List item = null;
try {
    item = upload.parseRequest(request);
} catch (FileUploadException e) {
    e.printStackTrace();
}
byte[] data = null;
Iterator iter = item.iterator();
while (iter.hasNext()) {
    FileItem itemm = (FileItem) iter.next();
    if (itemm.isFormField()) {
        String name = itemm.getFieldName();
        String value = itemm.getString();
        if (name.equalsIgnoreCase("umail23")) {
            umail = value;
            System.out.println("umail is  : " + umail);
        } else if (name.equalsIgnoreCase("fname")) {
            fname = value;
            System.out.println("fname is : " + fname);
        } else if (name.equalsIgnoreCase("lname")) {
            lname = value;
            System.out.println("lname is : " + lname);
        } else if (name.equalsIgnoreCase("dteobth")) {
            DOB = value;
            System.out.println("DOB is  : " + DOB);
        } else if (name.equalsIgnoreCase("uadds")) {
            Adds = value;
            System.out.println("Adds is : " + Adds);
        } else if (name.equalsIgnoreCase("uacity")) {
```

```
            city = value;
            System.out.println("city is : " + city);
        } else if (name.equalsIgnoreCase("ushname")) {
            sname = value;
            System.out.println("sname is : " + sname);
        } else if (name.equalsIgnoreCase("ushplce")) {
            splace = value;
            System.out.println("splace is : " + splace);
        } else if (name.equalsIgnoreCase("shpic")) {
            data = itemm.get();
            fileName = itemm.getName();
        } else {
            System.out.println(name);
        }
    } else {
        data = itemm.get();
        fileName = itemm.getName();
        System.out.println("File name is .....\t" + fileName);
    }
}
DB fb = new DB();
MnubarDet mb = new MnubarDet();
String usrname = mb.username(umail);
String uuid = mb.userID(umail);
saveFile = fileName;
String path = request.getSession().getServletContext().getRealPath("/");
String patt = path.replace("\\build", "");
String np = patt + "\\SHOP\\";
String strPath = np + saveFile;
File ff = new File(strPath);
```

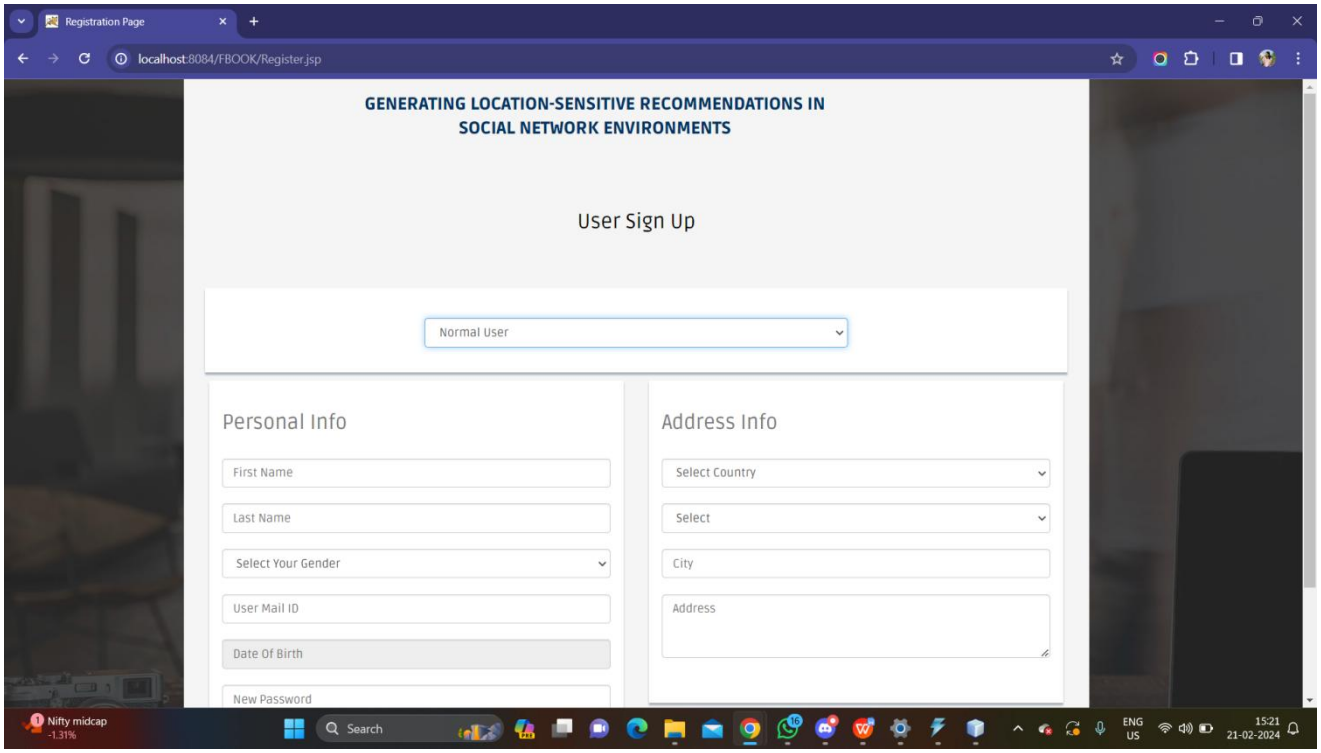
```
System.out.println("Fikle PATH\t" + strPath);
FileOutputStream fileOut = new FileOutputStream(ff);
fileOut.write(data, 0, data.length);
fileOut.flush();
fileOut.close();
System.out.println(saveFile);
File image = new File(strPath);
FileInputStream fis = new FileInputStream(image);
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/social", "root", "admin");
PreparedStatement pstmnt = con.prepareStatement("insert into storedetls(SH_Name,SH_CPlace,SH_LFlName,SH_Picture,SH_RUname,SH_RUMail,Usr_ID) values(?,?,?,?,?,?,?)");
pstmnt.setString(1, sname);
pstmnt.setString(2, splace);
pstmnt.setString(3, fileName);
pstmnt.setBinaryStream(4, (InputStream) fis, (int) (image.length()));
pstmnt.setString(5, usrname);
pstmnt.setString(6, umail);
pstmnt.setString(7, uuid);
int tr = pstmnt.executeUpdate();
if (tr > 0) {
    session.removeAttribute("ULOCATION");
    session.setAttribute("ULOCATION", city);
    int up = gf.Update("update users set F_Name='" + fname + "',L_Name='" + lname + "',DOB='" + DOB + "',City='" + city + "',Address='" + Adds + "' where Mail_ID='" + umail + "'");
    session.setAttribute("fbmsg", "Registered SuccessFully");
    response.sendRedirect("POWProfile.jsp");
```

```
            } else {
                session.setAttribute("fbmsg", "ERROR TRY AGAIN");
                response.sendRedirect("POBasicInfo.jsp");
            }

        } catch (Exception e) {
            System.out.println(e);
            e.printStackTrace();
        } finally {
            out.close();
        }
    }
    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
    /**
     * Handles the HTTP
     * <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
```

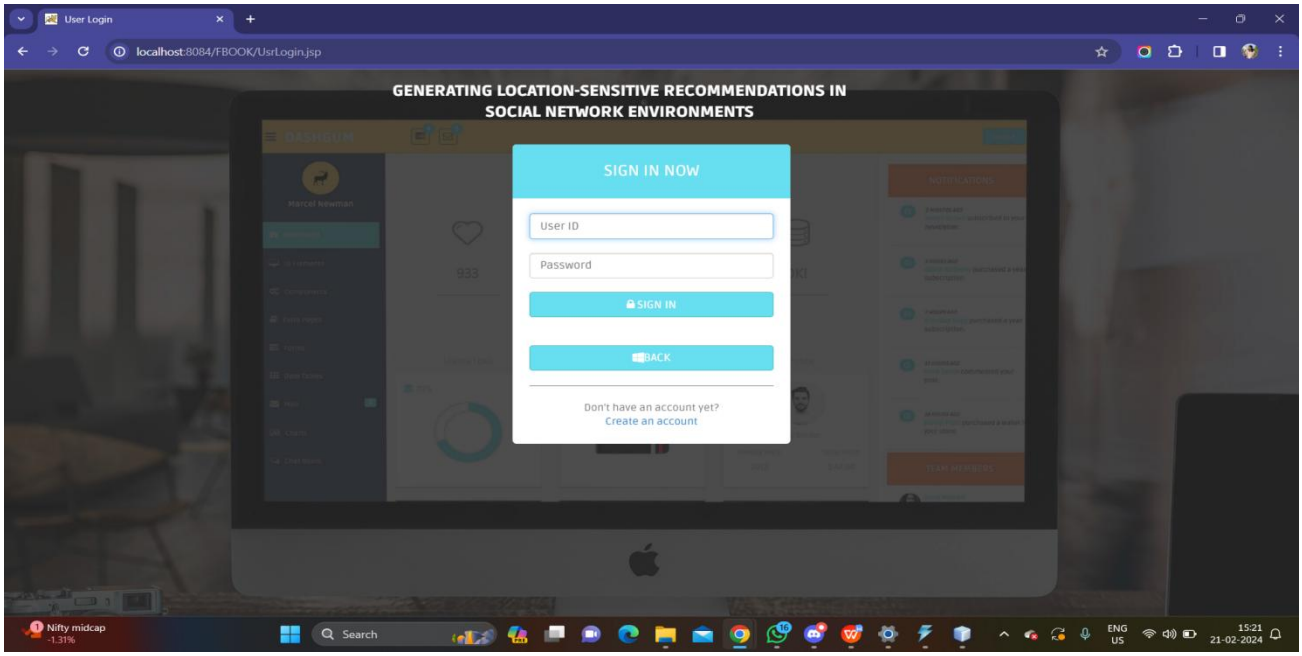
```
            Logger.getLogger(POUpdateProfile.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }
    /**
     * Handles the HTTP
     * <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        try {
            processRequest(request, response);
        } catch (SQLException ex) {
            Logger.getLogger(POUpdateProfile.class.getName()).log(Level.SEVERE, null,
ex);
        }
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
```

```
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>
}
```

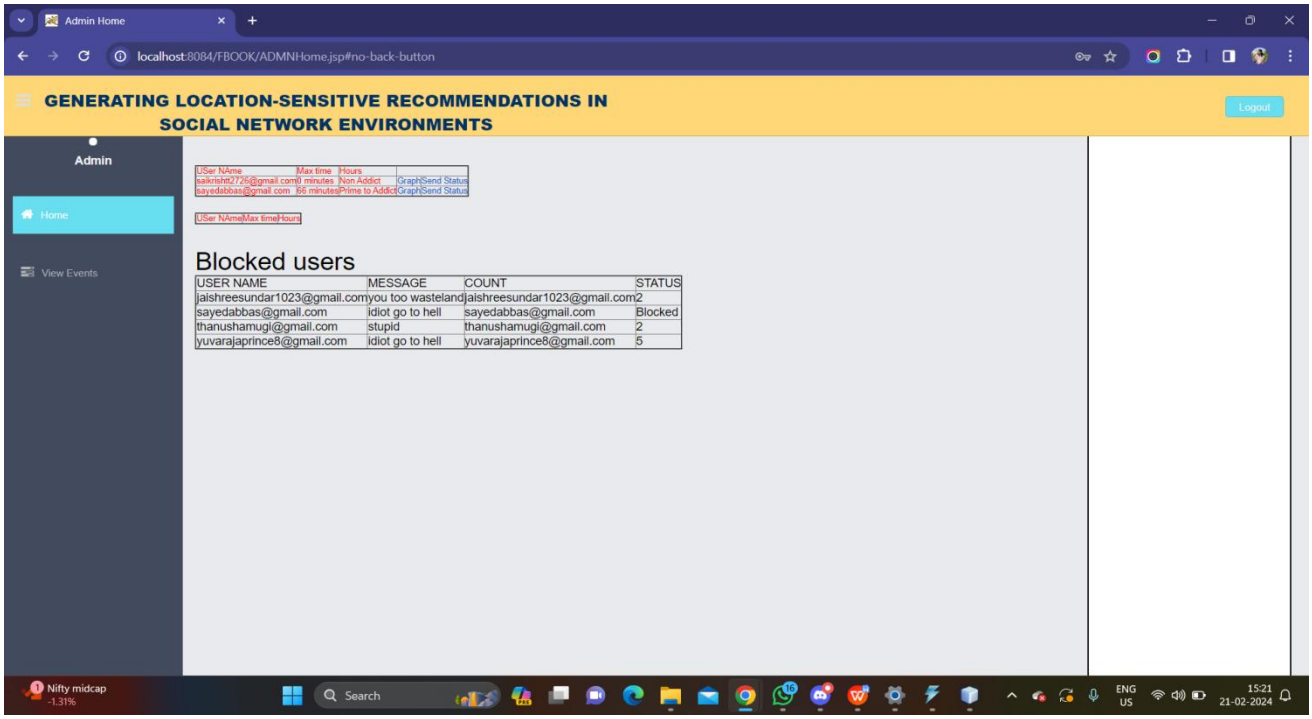
6.2 SCREENSHOTS



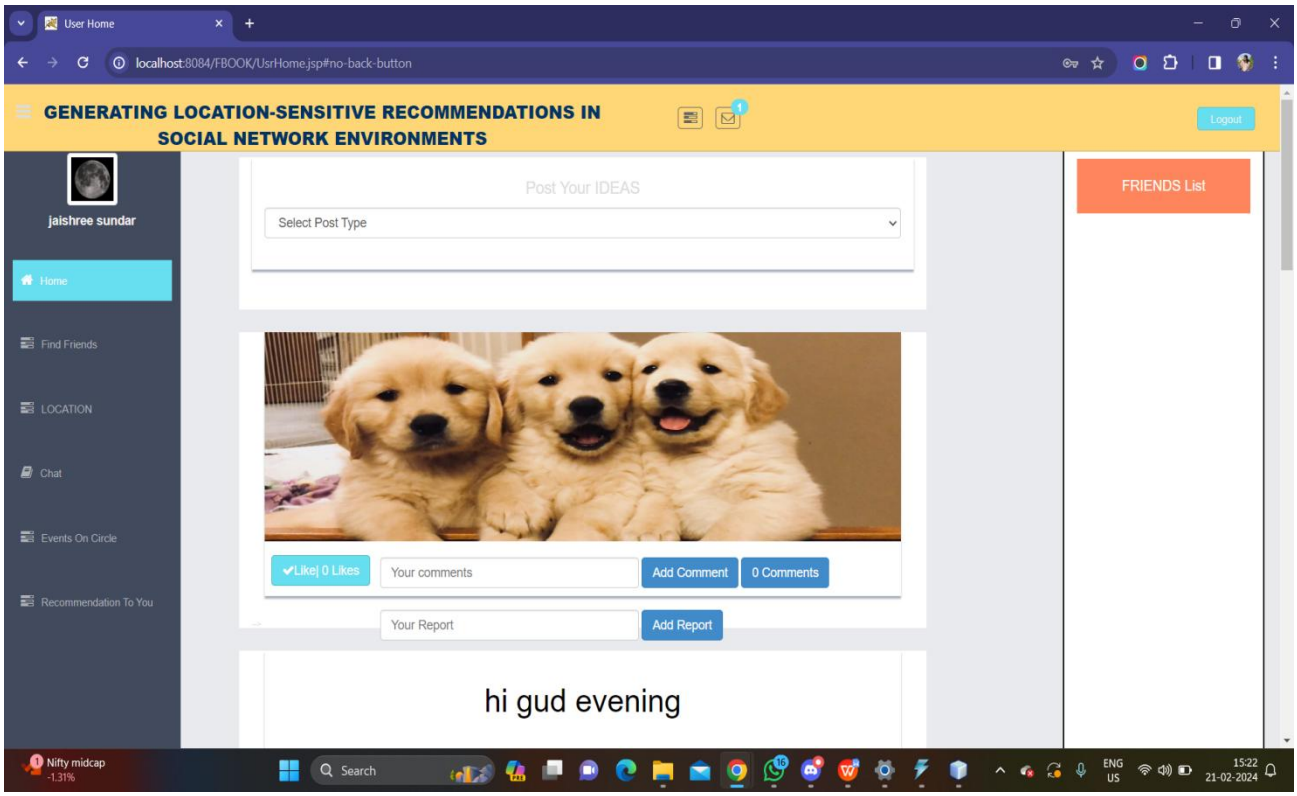
1.REGISTRATION PAGE



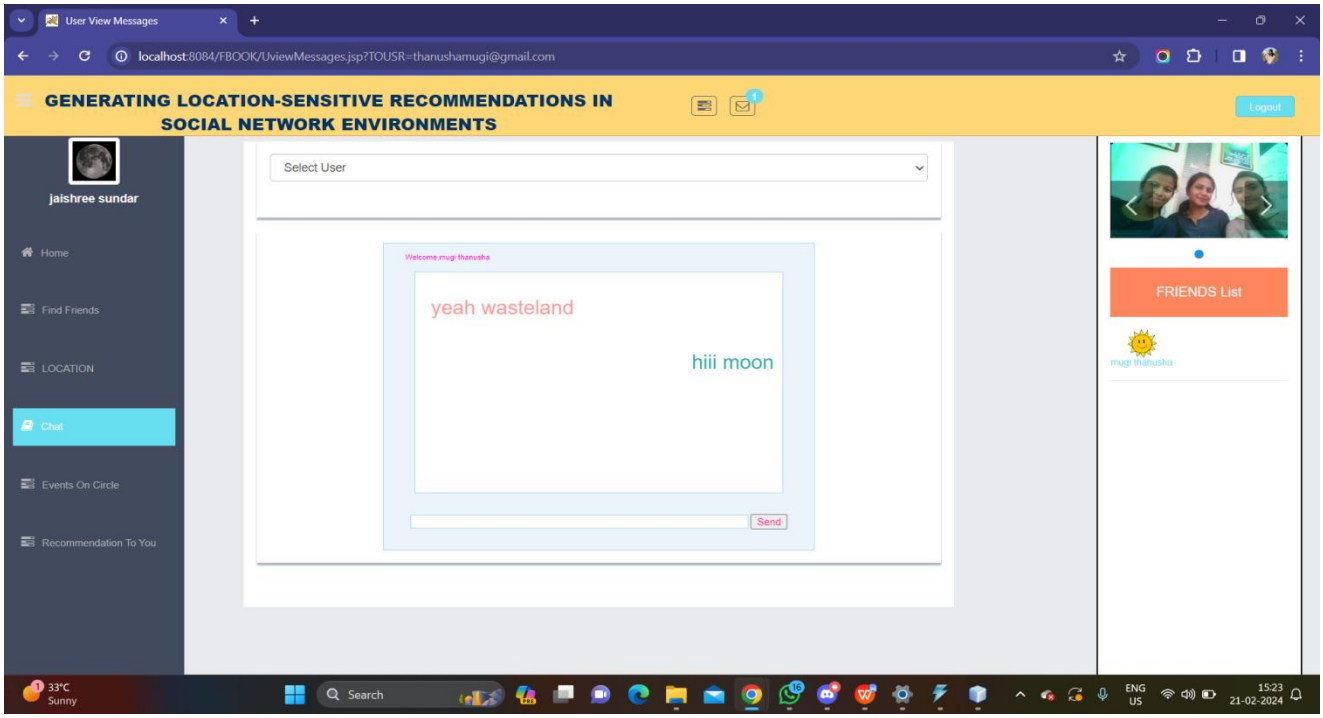
2.LOGIN PAGE



3. ADMIN PAGE



4. USER: HOME PAGE



5. USER: CHAT PAGE

REFERENCES

[1] Anuradha Khattar 1,2 And S. M. K. Quadri, Camm: Cross-Attention Multimodal Classification of Disaster-Related Tweets, Volume 10, 2022.

[2] Antonius Rachmat Chrismanto 1,2, Anny Kartika Sari 1, And Yohanes Suyanto, Enhancing Spam Comment Detection on Social Media With Emoji Feature and Post-Comment Pairs Approach Using Ensemble Methods of Machine Learning, Volume 11, 2023.

[3] Zhirui Luo , Qingqing Li , And Jun Zheng , (Member, IEEE), Deep Feature Fusion for Rumor Detection on Twitter, VOLUME 9, 2021.

[4] Abdulqader M. Almars , Malik Almaliki1, Talal H. Noor , Majed M. Alwateer , And Elsayed Atlam,, Hann: Hybrid Attention Neural Network for Detecting Covid-19 Related Rumors, VOLUME 10, 2022.

[5] Nazeeh Ghatasheh , Ismail Altaharwa , And Khaled Aldebei , Modified Genetic Algorithm for Feature Selection and Hyper Parameter Optimization: Case of XGBoost in Spam Prediction, VOLUME 10, 2022.

[6] Liu, H. Lu, and A. Nayak, ‘‘A spam transformer model for SMS spam detection,’’ IEEE Access, vol. 9, pp. 80253–80263, 2021.

[7] K. Chowdhury, ‘‘Spam identification on Facebook, Twitter and email using machine learning,’’ Central Eur. Res. J., vol. 6, no. 1, pp. 18–26, 2020.

[8] B. Alatas and H. Bingol, ‘‘Comparative assessment of light-based intel- ligent search and optimization algorithms,’’ Light Eng., vol. 28, no. 6, pp. 51–59, 2020.

[9] S. Kaddoura, G. Chandrasekaran, D. Elena Popescu, and J. H. Duraisamy, ‘‘A systematic literature review on spam content detection and classification,’’ PeerJ Comput. Sci., vol. 8, p. e830, Jan. 2022.

[10] Z. Wang and Z. Lin, ‘‘Optimal feature selection for learning-based algorithms for sentiment classification,’’ Cogn. Comput., vol. 12, no. 1, pp. 238–248, Jan. 2020.

[11] F. Concone, F. De Vita, A. Pratap, D. Bruneo, G. L. Re, and S.K. Das, “A fog-assisted system to defend against sybils in vehicular crowdsourcing,” Pervasive Mobile Comput., vol. 83, 2022, Art. no. 101612.

[12] B. Hanczar and A. Bar-Hen, “CASCARO: Cascade of classifiers for minimizing the cost of prediction,” Pattern Recognit. Lett., vol. 149, pp. 37–43, 2021.

[13] G. Lingam, R. R. Rout, D. Somayajulu, and S. K. Das, “Social bot_x0002_net community detection: A novel approach based on behavioral similarity in twitter network using deep learning,” in Proc. 15th ACM Asia Conf. Comput. Commun. Secur., 2020, pp. 708–718.

[14] Y. Wu, D. Lian, Y. Xu, L. Wu, and E. Chen, “Graph convolutional networks with Markov random field reasoning for social spammer detection,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 1054–1061.

[15] D. N. Koggalahewa, Y. Xu, and E. Foo, “Spam detection in social networks based on peer acceptance,” in Proc. Australas. Comput.Sci. Week Multiconference, 2020, Art. no. 8.